
# The professional JSON editor with OSCAL integration

**Document version:** 0.4.0
**Date:** 27 April 2026
**Licence:** AGPL-3.0
**OSCAL version:** 1.2.1
**MCP server:** v2.6.0
**Platforms:** Windows, Linux

# Table of Contents

# 1. Executive summary

**mjEdit** is more than just an editor – it is the first and currently only **open-source GUI application for editing OSCAL JSON files**, offering a professional and fully integrated working environment for Information Security Management Systems (ISMS) and, more generally, for work in the area of Governance, Risk & Compliance.

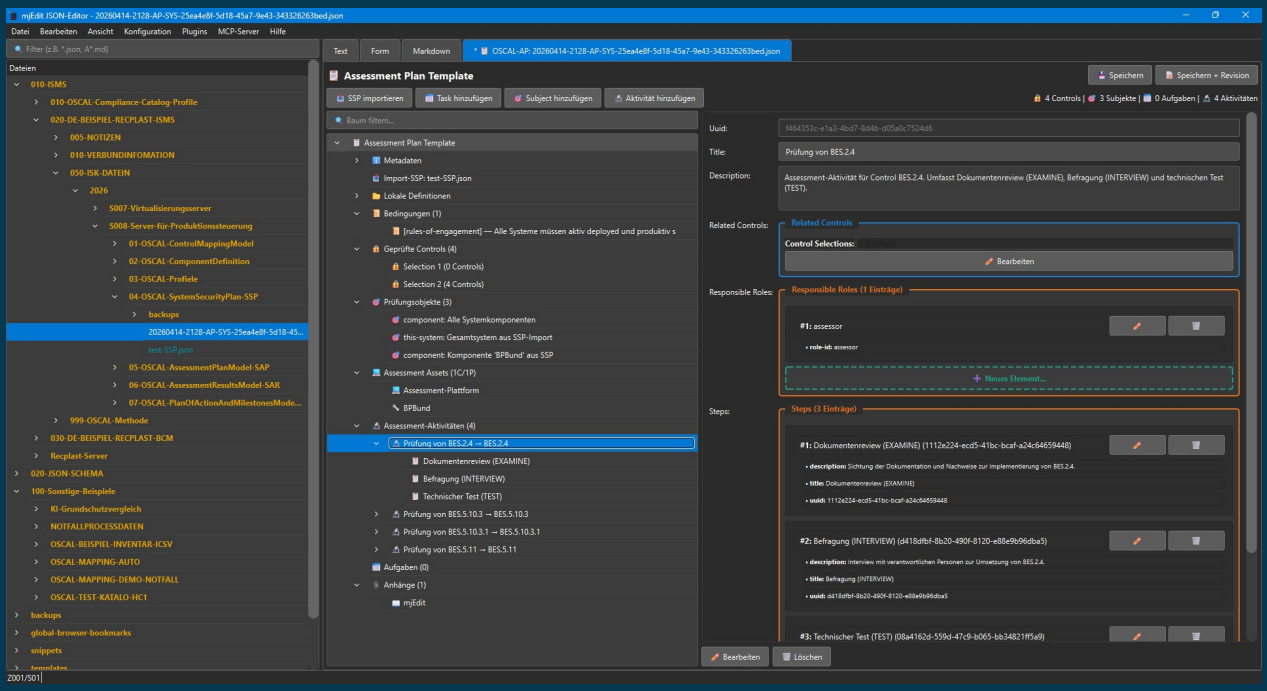


Figure 1: mjEdit – editing an OSCAL SSP document

For **ISMS practitioners** this means: finally a tool that not only renders complex JSON structures transparently but also eases daily documentation work with dynamic forms (qFORM). **Consultants** benefit from the multi-tab interface that allows them to author policies in Markdown, review compliance documents and work directly in the PDF viewer with annotations or redactions – all at the same time. **Auditors** receive a central platform where they can analyse JSON data, accompanying documentation and web content in the integrated browser side by side – no media breaks, no tool switching.

mjEdit thus combines the most important tools for daily work in a single interface:

- **JSON editor** for OSCAL and other structured data
- **qFORM system** for fast, consistent data entry
- **Markdown editor with live preview** for policies and reports
- **PDF viewer with annotations and redaction** for audit-proof documentation

- **Integrated web browser** for research and direct source linking

The result: a seamless, efficient and professional environment covering the entire workflow of ISMS editing, consulting and auditing – and this as a freely available open-source solution.

**Key features at a glance:**

- **JSON editor** with syntax highlighting, real-time validation, auto-repair, regex search, code folding and intelligent performance management for files up to 1 MB+
- **qFORM form system** – JSON arrays automatically rendered as editable forms with text fields, checkboxes, dropdowns, date pickers and embedded QC scripts (Python sandbox)
- **Markdown editor** with split view, GitHub Flavored Markdown, extended task system (priorities, assignment, time tracking) and PDF / HTML export
- **PDF viewer** with annotations (highlighter, notes, freehand), secure text redaction and zoom up to 400 %
- **Browser tab** – Chromium-based web browser with bookmarks, tabs and lazy loading
- **Plugin system** – extensible via hook-based plugins with tab registration and menu integration
- **Cross-cutting:** efficient tree view with real-time filter, automatic backups, version archive, 4 themes (Dark / Light / Blue / Green), full bilingualism (de / en)

**OSCAL compliance – the unique selling proposition:**

- **All 8 OSCAL document types** (v1.2.1) fully supported – from catalog to mapping collection, with specialised editors per document type
- **AI integration with 88 MCP tools** for AI-assisted automation – create, validate and automate OSCAL documents via AI agents (Claude, Cursor, VS Code Copilot, AnythingLLM)
- **Pydantic-based validation** – schema-compliant documents from the moment of creation
- **BSI IT-Grundschutz** and **NIST SP 800-53** pre-installed
- **Network discovery** – automatically import IP addresses, hostnames and MAC addresses into the SSP inventory

## 1.1 What sets mjEdit apart from other JSON editors

| Feature | Standard JSON editors | mjEdit |
|---|---|---|
| OSCAL detection | Manual | Automatic – detects document type and opens the specialised tab |
| Schema validation | Generic | OSCAL v1.2.1 schemas pre-installed, real-time validation |
| AI integration | None | 88 MCP tools for fully automated OSCAL workflows |
| Compliance data | Not included | BSI Grundschutz (2,128 controls), NIST 800-53 (468 controls) |
| Document chains | Not possible | One click: Profile → SSP → AP → AR → POA&M |
| Inventory management | Not included | OOP analogy: components (classes) ↔ inventory items (instances) |

# 2. Target audiences and use cases

## 2.1 Information security officers (ISO / CISO)

**Daily work:** Maintaining ISMS documentation, implementing controls, preparing audits.

**How mjEdit helps:**

- Ready-made BSI Grundschutz and NIST catalogs as a starting point
- Profile tailoring: select only the controls relevant to your organisation
- SSP generation: derive the system security plan directly from the profile
- Assessment planning: generate assessment plans from the SSP
- POA&M tracking: track remediation actions with deadlines and owners

## 2.2 Compliance auditors

**Daily work:** Reviewing controls, collecting evidence, writing reports.

**How mjEdit helps:**

- Document assessment results directly in the editor
- Link findings with severity and evidence
- Markdown export for auditable reports
- Schema validation guarantees standard-compliant documents
- Reverse-lookup: find all inventory items associated with a component

## 2.3 IT architects and system administrators

**Daily work:** Documenting systems, maintaining network topologies, patch management.

**How mjEdit helps:**

- **Network discovery:** automatically import IP addresses, hostnames and MAC addresses into the SSP inventory
- **Component library:** define software, hardware and services as reusable building blocks
- **Inventory management:** link concrete server instances with components
- **CSV import / export:** connect to existing asset management systems
- **Network diagrams:** automatic NWDiag generation from inventory data

## 2.4 DevSecOps teams and AI developers

**Daily work:** Security-as-code, automated compliance pipelines, AI-assisted workflows.

**How mjEdit helps:**

- **88 MCP tools** for programmatic OSCAL control
- **Batch generation:** 8 servers × complete document chain in a single MCP call
- **execute_steps:** up to 20 tool calls in a single request
- **Claude, Cursor, VS Code Copilot, AnythingLLM:** direct integration into existing development environments
- **AnythingLLM RAG:** build a compliance knowledge base with your own documents and link it to mjEdit via MCP
- **Pydantic validation:** model-based data integrity for CI/CD pipelines

---

# 3. OSCAL – why mjEdit makes the difference

## 3.1 All 8 OSCAL document types

mjEdit supports the full OSCAL v1.2.1 standard:

| Document type | Short | Function | mjEdit feature |
|---|---|---|---|
| **Catalog** | catalog | Define control framework | Browse, edit and create groups / controls |
| **Profile** | profile | Select baseline | Select controls, tailoring, set parameters |
| **System Security Plan** | ssp | Document the system | Components, inventory, control implementation |
| **Assessment Plan** | ap | Plan the assessment | Define assessment methods, scope, timeline |
| **Assessment Results** | ar | Document results | Record findings, observations and evidence |
| **Plan of Action & Milestones** | poam | Track remediation | Remediation items, deadlines, status tracking |
| **Component Definition** | component-definition | Define building blocks | Reusable component libraries |
| **Mapping Collection** | mapping | Map frameworks | Bidirectional control mapping between standards |

## 3.2 The OOP principle: components as classes, inventory as instances

mjEdit maps the OSCAL architecture intuitively – analogous to object-oriented programming:

**Component = class (blueprint)**

```
"Apache HTTP Server" – type: Software, description: Web server
```

**Inventory item = instance (concrete asset)**

```
"Apache 2.4.58 on server web-01" – IP: 192.168.1.10, FQDN: web-01.example.com
```

**The linkage:**

- Open an inventory-item dialog in the SSP tab
- Select the relevant software / hardware building blocks via multi-select in tab 3 “Components”
- mjEdit automatically creates the `implemented-components` reference with `component-uuid`

**Automation features:**

- **Detect orphaned items:** inventory items without a component link are automatically flagged in the SSP inventory tab – unlinked assets can be filtered and assigned to a component directly
- **Reverse-lookup:** display all concrete server instances from a component
- **Duplicate protection:** duplicate links are automatically prevented
- **CSV import:** import inventory lists from Excel or CMDB

## 3.3 Automatic OSCAL document chains

A key advantage of mjEdit: you do not have to create documents one by one. mjEdit can automatically generate the entire compliance chain – from a catalog through profile and SSP all the way to assessment plan, results and POA&M.

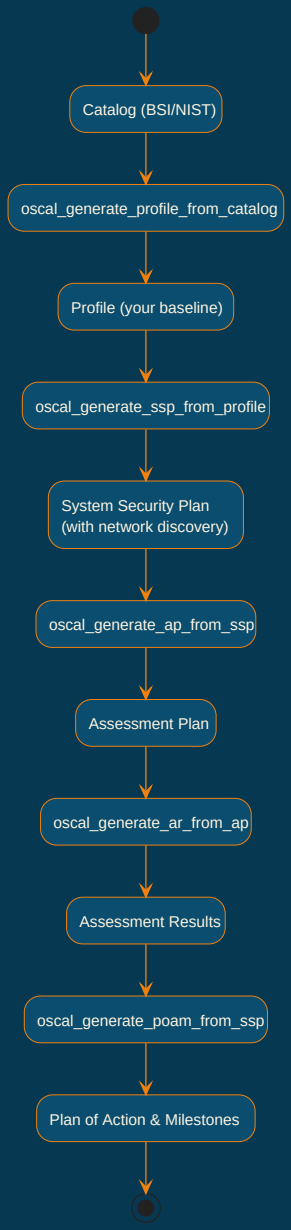


Figure 2: Diagramm

**Each step:**

- Carries over UUID references automatically
- Sets metadata (title, version, last-modified) correctly
- Validates against OSCAL v1.2.1 schema
- Supports custom prose and assessment actions

## 3.4 Batch generation for server landscapes

For organisations with multiple systems, mjEdit offers **batch server chain generation**:

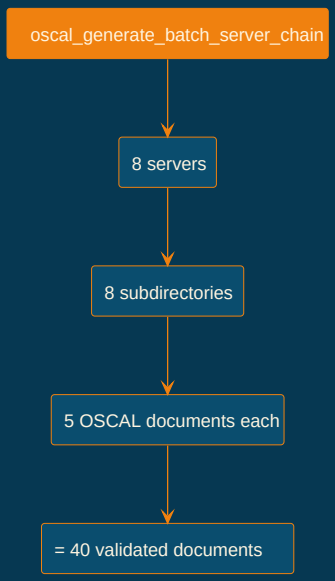


Figure 3: Diagramm

Each server receives:

- Its own profile with system-specific controls
- SSP with network inventory (IP, hostname, MAC)
- Assessment plan with adapted assessment methods
- Assessment results as a template
- POA&M for remediation tracking

## 3.5 Evidence Workflow with Status Tracking (NEW 05/2026)

Since 16 May 2026, mjEdit visualises the complete 3-stage evidence workflow **SSP → AP → AR** directly in the TreeView. Auditors and ISMS owners see at a glance which evidence has already been reviewed and what is still outstanding.

**Prominent tree nodes in every document:**

- **SSP:** `Evidence Links` per `implemented-requirement` – which evidence is collected for this control?
- **AP:** `Planned Evidence` (top-level) – which evidence has to be reviewed? Filtered via the OSCAL property `validation-status`.
- **AR:** `AP Evidence to Review (X/Y reviewed)` – with live status ☑ (reviewed) or ⏳ (pending) per AP-planned item.

**One-click observation:**

Right-click on a ⏳ item in the AR tab → **„📋 Create observation for this evidence"**. mjEdit creates an OSCAL v1.2.1-compliant observation with `methods=["EXAMINE"]` and `relevant-evidence[0].href = "#<uuid>"` as a schema-compliant anchor referencing the AP back-matter resource. Optionally `props.name=evidence-source` with value `"ap"` can be set as a source marker.

**Cross-document linking:**

When opening an AR document, mjEdit automatically loads the linked assessment plan via `import-ap.href` and reconciles planned evidence against referenced observations.

**Benefits:**

- Seamless traceability plan → review → evidence
- Instant status overview without manual list maintenance
- 100% OSCAL v1.2.1-compliant – exportable to external GRC tools
- Significantly reduces audit preparation effort

## 3.6 Profile tailoring and mapping

**Profile tailoring:**

- Select controls via the catalog tree (with cascade selection for entire groups)
- Set parameter values (e.g. password length, audit interval)
- Alterations: add to, restrict or replace controls
- Profile resolution: generate a flat catalog from profile + base catalog

**Mapping collection:**

- Map controls between different frameworks using automatic AI assignment (e.g. BSI ↔ NIST)

- Bidirectional format adapter (OSCAL-NIST ↔ internal mjEdit format with automatic validation)
- Quick overview through visual highlighting of linked controls
- Markdown export of the complete mapping table

## 3.7 Schema validation and data quality

mjEdit validates on three levels:

| Level | Mechanism | When |
|---|---|---|
| **1. Pydantic models** | Python-based data validation | At creation and editing |
| **2. JSON Schema** | NIST OSCAL v1.2.1 schemas | On request or before saving |
| **3. Reference integrity** | UUID / href / control-ID check | When validation is triggered |

**Pre-installed schemas:**

- oscal_catalog_schema.json
- oscal_profile_schema.json
- oscal_ssp_schema.json
- oscal_component_schema.json
- oscal_assessment-plan_schema.json
- oscal_assessment-results_schema.json
- oscal_poam_schema.json

## 3.8 Network discovery – automatic IT inventory for your SSP

The **Network Discovery plugin** scans the local network via ARP sweep and transfers the discovered hosts as OSCAL v1.2.1-compliant `inventory-item` entries directly into an open SSP tab. It closes the most time-consuming gap in every SSP project: the manual capture of the actually existing IT inventory.

> **Note:** The plugin is disabled by default and must be consciously activated in the Plugin Manager. On Windows, Npcap is additionally required as a driver.

| Role | Added value |
|---|---|

## Added value by role

| Role | Added value |
|---|---|
| **ISO / CISO** | Up-to-date, complete inventory without manual capture effort; asset drift becomes visible immediately with every scan |
| **Compliance auditor** | Reproducible scan with timestamp as a revision-proof audit record; direct comparison between scanned actual state and documented target inventory |
| **IT architect** | Network topology automatically as OSCAL inventory – foundation for network diagrams and component linkages |
| **DevSecOps** | MCP-assisted automation: `oscal_generate_ssp_from_profile` optionally triggers a discovery scan and populates the inventory without manual input |

## How to use it

1. Activate the plugin in the Plugin Manager (*Extras → Plugin Manager*)
2. On Windows: install Npcap (raw socket access)
3. Open an SSP document in mjEdit (the SSP tab must be active)
4. Open menu **Plugins → 🔎 Network Discovery...**
5. Enter subnet in CIDR notation (e.g. `192.168.1.0/24`)
6. Set the timeout and click **Start scan** (duration: ~30 seconds for 256 hosts)
7. Select / deselect hosts in the results table; optionally activate "Skip duplicates"
8. Choose the target SSP from the dropdown, then click **"Add to SSP"**

## Results and OSCAL target document

The scan results are stored as `inventory-item` entries in the section `system-implementation.inventory-items` of the active **System Security Plan (SSP)**:

| Scan result | OSCAL field |
|---|---|
| IP address | `props[].name="ipv4-address"` |
| MAC address | `props[].name="mac-address"` |

| Scan result | OSCAL field |
|---|---|
| FQDN (fully qualified domain name) | `props[].name="fqdn"` |
| Hostname (short form) | `props[].name="hostname"` |
| Scan timestamp | `props[].name="discovered-at"` |

Each entry automatically receives an RFC 4122-compliant UUID, an empty `description` field for manual annotation of the asset function, and an empty `responsible-parties` list for later ownership assignment. The populated SSP document is marked as modified (`*` in the tab title) and can be saved with **Ctrl+S**.

**Security:**

- No external service, no cloud connection – all data stays within the local network
- Subnet limited to a maximum of 4,096 IP addresses (anti-DoS protection)
- Confirmation dialog with the note "Only scan your own networks" before every scan

**Practical example:** A consultant opens a prepared SSP tab, triggers a scan on `192.168.50.0/24` (duration approx. 30 seconds), receives a list of 47 hosts including hostnames from the internal DNS, selects 41 production systems (6 test VMs are excluded) and imports them into the SSP with a single click. The `description` and `responsible-parties` fields are then completed per host in the SSP tab – the asset inventory is built in **under 15 minutes**, fully OSCAL-compliant and audit-proof.

---

## 3.9 Automatic standard network architecture from IT inventory

As soon as an SSP contains a populated `inventory-items` array – whether through manual entry, CSV import or the network discovery scan – mjEdit can automatically generate an **NWDiag network diagram**. The diagram shows the captured assets in a standardised, documentation-ready layer-3 topology format and is embedded directly as an OSCAL-compliant artefact in the SSP document.

### Added value by role

| Role | Added value |
|---|---|
| **ISO / CISO** | Visual system boundary overview without additional tools; always synchronised with the current SSP inventory – no more outdated Visio documents |
| **Compliance auditor** | Network diagram as an OSCAL-compliant record of the captured system boundaries (authorisation boundary) in the SSP; derived directly from the primary data, not manually created |

| Role | Added value |
|---|---|
| **IT architect** | Automatically derived baseline documentation as a starting point for more detailed architecture diagrams; subnet grouping and component types are detected automatically |
| **DevSecOps** | Diagram generation via MCP tool `oscal_generate_ssp_from_profile` or directly via MCP call for pipeline integration; SVG export for CI/CD documentation pipelines |

## How to use it

The network architecture is generated directly from the open SSP tab:

1. Open an SSP with a populated inventory in the SSP tab
2. In the SSP tab → context menu or toolbar button **"Generate network diagram"**
3. mjEdit reads all `inventory-items` with IP addresses and groups them automatically by subnet
4. An NWDiag script is generated and rendered as a diagram
5. Choose the output format: SVG, PNG or direct embedding in OSCAL back matter

## Results and OSCAL target document

The generated diagram is embedded in the **System Security Plan (SSP)**:

| Artefact | OSCAL target |
|---|---|
| NWDiag source code | `back-matter.resources[]` with `media-type: "text/plain"` |
| Diagram image (SVG) | `back-matter.resources[]` with `media-type: "image/svg+xml"` |
| Diagram image (PNG) | `back-matter.resources[]` with `media-type: "image/png"` |
| Diagram reference | `system-characteristics.network-architecture.diagrams[]` |

## What is detected and rendered automatically?

| Detected element | Data source in OSCAL inventory |
|---|---|
| **Subnets / network segments** | IP addresses of `inventory-items` (property `ipv4-address`) |
| **Hosts** | All `inventory-items` with an IP address set |

| Detected element | Data source in OSCAL inventory |
|---|---|
| **Hostnames / labels** | Property `hostname` or `fqdn` |
| **Component type** | Component linked to the `inventory-item` (server, switch, firewall, etc.) |
| **Network segments** | Automatic grouping by /24 subnet |

## Practical example

An SSP contains 47 hosts in two subnets after the network discovery scan (`10.0.1.0/24` server segment, `10.0.2.0/24` client segment). With one click mjEdit generates a diagram that:

- renders both segments as NWDiag network groups,
- lists hosts with their hostnames,
- annotates the associated component types (e.g. “Ubuntu 24.04”, “Windows 11”).

**The result:** A complete, up-to-date and OSCAL-compliant network architecture documentation – automatically from the data already present in the SSP.

> **Note:** The diagram shows a simplified layer-3 topology (subnet view). For complete network diagrams with VLAN configuration, firewall rules and protocol details, we recommend dedicated network management tools as a complement.

---

# 4. AI integration via MCP – your compliance on autopilot

> **IMPORTANT NOTE**: The MCP integration is at an early stage of development. 88 tools are currently implemented, covering a broad range of functions – from document creation through validation to GUI control. In the coming months further tools and resources will be added, or some may be removed, to continuously improve automation capabilities. At this time it is not advisable to build a production integration on this development state. The MCP tools are fully functional and can be tried in test environments with supported AI agents (Claude Desktop, Cursor, VS Code Copilot, AnythingLLM). Please refer to the documentation and tool descriptions for details on available functions and their use cases.

## 4.1 What is MCP?

The **Model Context Protocol (MCP)** is an open standard that allows AI systems to communicate with external tools. mjEdit implements a complete MCP server with two transport options:

- **STDIO:** Direct process communication (for Claude Desktop, Cursor, VS Code Copilot)
- **SSE:** HTTP Server-Sent Events on localhost (for AnythingLLM Docker, custom integrations)

## 4.2 88 MCP tools in 15 categories

mjEdit stands out as an integrated working environment – not only through its unique specialisation in **OSCAL JSON documents** but also through its support for **88 MCP tools in 15 categories**, which provides a range of functions that goes far beyond simple file editing.

These tools are not merely technical helpers; they form the foundation for an **AI-assisted way of working** that directly supports ISMS practitioners, consultants and auditors in their daily workflow. Whether quickly creating and validating OSCAL documents, transforming JSON structures, interactively editing Markdown policies or annotating PDF documents in an audit-proof way – mjEdit combines all steps in one interface.

Particularly noteworthy are the **guided workflows with 15 MCP prompts**, which simplify complex tasks such as creating complete OSCAL document chains, performing compliance checks or interactively tailoring profiles. This makes mjEdit a true **productivity hub** that not only saves time but also ensures the quality and consistency of work.

## File operations (12 tools)

| Tool | Description |
|---|---|
| `file_create` | Create a file (with batch support for large files) |
| `file_open` | Open a file with automatic tab routing |
| `file_read` | Read a file without opening it in the editor |
| `file_write` | Write a file directly to disk |
| `file_save` | Save the active file with backup |
| `file_save_as` | Save under a new path |
| `file_close` | Close the active file |
| `file_append` | Incremental writing for large files |
| `file_list` | List files with a glob pattern |
| `file_get_active` | Path + content of the current file |
| `dir_create` | Create a directory |
| `dir_rename` | Rename a directory |

## JSON operations (6 tools)

| Tool | Description |
|---|---|
| `json_get_content` | Retrieve JSON content from the editor |
| `json_set_content` | Set JSON content (with validation) |
| `json_format` | Pretty-print with configurable indent |
| `json_validate` | Syntax validation with line / column information |
| `json_export` | Export as XML or CSV |
| `json_query` | Query values via dot-path (e.g. `metadata.title`) |

## Content manipulation (3 tools)

| Tool | Description |
|---|---|
| `content_search` | Text / regex search with context lines |
| `content_replace` | Text replacement with regex and capture groups |
| `content_insert_at` | Insert content at a specific line |

## Markdown operations (8 tools)

| Tool | Description |
|---|---|
| `markdown_get_content` | Retrieve Markdown content |
| `markdown_set_content` | Set Markdown content |
| `markdown_insert_snippet` | Insert a snippet at the cursor position |
| `markdown_replace_section` | Replace a section by heading |
| `markdown_get_toc` | Generate table of contents |
| `markdown_export_to_html` | HTML export |
| `markdown_export_to_pdf` | PDF export |
| `markdown_format` | Format Markdown |

### qFORM forms (9 tools)

| Tool | Description |
|---|---|
| `qform_create` | Create a new form with field types |
| `qform_get_data` | Retrieve structured form data |
| `qform_set_field` | Set field value (by label or index) |
| `qform_add_field` | Add a field with position control |
| `qform_remove_field` | Remove a field |
| `qform_render` | Render the form in the GUI |
| `qform_execute_qc_script` | Execute a QC script in the sandbox |
| `qform_create_qc_script` | Create a QC script in CRC32:Base64 format |
| `qform_list_fields` | List all fields with types and values |

### Validation (5 tools)

| Tool | Description |
|---|---|
| `validate_json_schema` | Validate JSON against a schema |
| `validate_oscal_document` | Check against the official OSCAL schema |
| `validate_oscal_references` | Check UUID / href / control-ID references |
| `validate_qform` | Check qFORM structure and consistency |
| `validate_file_integrity` | Check JSON syntax, encoding and file size |

| Tool | Description |
|---|---|

## OSCAL creation (8 tools)

| Tool | Description |
|---|---|
| `oscal_create_catalog` | Create a catalog with auto-UUID / metadata |
| `oscal_create_profile` | Create a profile with import href |
| `oscal_create_ssp` | Create an SSP with system characteristics |
| `oscal_create_poam` | Create a POA&M with placeholder items |
| `oscal_create_assessment_plan` | Create an assessment plan with include-all |
| `oscal_create_assessment_results` | Create assessment results |
| `oscal_create_component_definition` | Create a component definition |
| `oscal_detect_type` | Automatically detect the OSCAL document type |

## OSCAL queries (8 tools)

| Tool | Description |
|---|---|
| `oscal_list_controls` | List all controls with IDs, titles and classes |
| `oscal_get_control` | Retrieve a single control with all details |
| `oscal_search` | Search an OSCAL document (text + regex) |
| `oscal_get_metadata` | Retrieve metadata of a document |
| `oscal_list_groups` | List catalog groups |
| `oscal_get_statistics` | Retrieve statistics (control count etc.) |
| `oscal_select_controls` | Select controls by ID in the GUI tab |
| `oscal_select_group` | Select an entire group in the GUI |

## OSCAL editing (7 tools)

| Tool | Description |
|---|---|
| `oscal_add_control` | Add a control to a catalog |
| `oscal_update_metadata` | Update metadata fields |
| `oscal_add_property` | Add a property via JSON path |
| `oscal_add_parameter` | Add a set-parameter |
| `oscal_add_back_matter_resource` | Add a back-matter resource |
| `oscal_update_implementation_status` | Set control status in the SSP |
| `oscal_add_role` | Add a role to the document |

## OSCAL control management (9 tools)

| Tool | Description |
|---|---|
| `oscal_update_control` | Update a control (title, class, parts, props) |
| `oscal_remove_control` | Remove a control from a catalog |
| `oscal_add_group` | Add a group to a catalog |
| `oscal_remove_group` | Remove a group with all controls |
| `oscal_profile_list_imports` | List profile imports with include / exclude |
| `oscal_profile_add_import` | Add an import to a profile |
| `oscal_profile_remove_import` | Remove an import by href |
| `oscal_profile_add_alter` | Add an alteration to a control (tailoring) |
| `oscal_profile_set_parameter` | Set a parameter value in the modify section |

## OSCAL generation (8 tools)

| Tool | Description |
|---|---|
| `oscal_generate_profile_from_catalog` | Profile from catalog with selected controls |
| `oscal_generate_ssp_from_profile` | SSP with network discovery support |
| `oscal_generate_ap_from_ssp` | Generate assessment plan from SSP |
| `oscal_generate_poam_from_ssp` | Generate POA&M from SSP |
| `oscal_generate_ar_from_ap` | Generate assessment results from plan |
| `oscal_generate_full_chain` | Complete chain without tailoring |
| `oscal_generate_tailored_chain` | Chain with assessment actions / custom prose |
| `oscal_generate_batch_server_chain` | Batch generation for multiple servers |

## OSCAL model API (7 tools – Pydantic-based)

| Tool | Description |
|---|---|
| `oscal_model_create` | OSCAL via Pydantic model with template support |
| `oscal_model_open` | Open an OSCAL file in the appropriate tab |
| `oscal_model_delete` | Delete OSCAL (with .bak backup) |
| `oscal_model_modify` | Batch operations (set / append / remove / replace) |
| `oscal_model_save` | Save model + GUI refresh |
| `oscal_model_query` | Query from model (paginated) |
| `oscal_model_list_templates` | List available OSCAL templates |

### Editor operations (8 tools)

| Tool | Description |
|---|---|
| `editor_undo` | Undo |
| `editor_redo` | Redo |
| `editor_cut` | Cut |
| `editor_copy` | Copy |
| `editor_paste` | Paste |
| `editor_select_all` | Select all |
| `editor_find_replace` | Find & replace dialog |
| `editor_go_to_line` | Navigate to line |

### Template management (5 tools)

| Tool | Description |
|---|---|
| `template_list` | List templates |
| `template_create` | Create a template |
| `template_delete` | Delete a template |
| `template_export` | Export a template |
| `template_import` | Import a template |

### GUI control (8 tools)

| Tool | Description |
|---|---|
| `execute_steps` | **Batch tool:** up to 20 tool calls in ONE request |
| `get_help` | Full tool list + workflows (bilingual) |
| `gui_show_message` | Display a message (info / warning / error) |
| `gui_show_tab` | Switch tab (index or name) |
| `gui_get_status` | Editor status + tool list with examples |
| `gui_set_theme` | Set theme (dark / light / blue / green) |
| `gui_list_tabs` | List all open tabs with type / path |
| `gui_get_tab_content` | Read the content of a specific tab |

## 4.3 22 MCP resources

AI agents can query contextual information directly:

| Resource | Description |
|---|---|
| `mjedit://config/app` | Current application configuration |
| `mjedit://config/keybindings` | Configured keyboard shortcuts |
| `mjedit://config/security` | Security settings |
| `mjedit://config/mcp` | MCP server configuration |
| `mjedit://schema/oscal/catalog` | OSCAL catalog schema v1.2.1 |
| `mjedit://schema/oscal/profile` | OSCAL profile schema v1.2.1 |
| `mjedit://schema/oscal/ssp` | OSCAL SSP schema v1.2.1 |
| `mjedit://schema/oscal/component` | OSCAL component schema v1.2.1 |
| `mjedit://schema/oscal/assessment-plan` | OSCAL assessment plan schema v1.2.1 |
| `mjedit://schema/oscal/assessment-results` | OSCAL assessment results schema v1.2.1 |
| `mjedit://schema/oscal/poam` | OSCAL POA&M schema v1.2.1 |
| `mjedit://snippets/json` | JSON code snippets |
| `mjedit://snippets/markdown` | Markdown snippets |
| `mjedit://snippets/oscal` | OSCAL snippets |
| `mjedit://snippets/qform` | qFORM snippets |
| `mjedit://snippets/qscript` | QC script templates |
| `mjedit://templates/list` | All available templates |
| `mjedit://editor/active-file` | Path + content of the current file |
| `mjedit://editor/active-tab` | Active tab (type, index, file) |
| `mjedit://plugins/list` | Active plugins |
| `mjedit://oscal/controls` | Controls of the current document |
| `mjedit://oscal/required-fields` | Required / optional fields per OSCAL type |

## 4.4 15 MCP prompts for guided workflows

| Prompt | Description |
|---|---|
| `oscal_create_catalog_prompt` | Catalog creation with guidance |
| `oscal_create_ssp_prompt` | SSP creation with system definition |
| `oscal_validate_prompt` | Interactive OSCAL validation |
| `oscal_compliance_check_prompt` | Compliance check against framework |
| `oscal_edit_controls_prompt` | Interactive control editing |
| `oscal_profile_tailoring_prompt` | Guided profile tailoring |
| `oscal_batch_edit_prompt` | Batch editing of controls |
| `oscal_network_discovery_prompt` | SSP network discovery |
| `oscal_batch_server_chain_prompt` | Batch document chains for servers |
| `oscal_migration_prompt` | OSCAL version migration |
| `qform_design_prompt` | Form design via AI |

| Prompt | Description |
|---|---|
| `qform_qc_script_prompt` | QC script creation via AI |
| `json_transform_prompt` | JSON transformation |
| `markdown_document_prompt` | Markdown document creation |
| `security_audit_prompt` | Security analysis |

## 4.5 AnythingLLM – the AI agent with a RAG knowledge base for compliance professionals

While Claude Desktop, Cursor and VS Code Copilot offer powerful MCP integration as cloud-based AI agents, **AnythingLLM** plays a special role: as a self-hosted, open-source platform, AnythingLLM combines the capabilities of an AI assistant with a **Retrieval-Augmented Generation (RAG)** knowledge base – entirely under your control, optionally local or in your own Docker container.

### What is AnythingLLM?

**AnythingLLM** is an all-in-one AI platform that combines the following core capabilities:

| Capability | Description |
|---|---|
| **Multi-LLM support** | OpenAI GPT-4o, Anthropic Claude, local models (Ollama, LM Studio), Azure OpenAI and others |
| **RAG engine** | Embed your own documents (PDF, DOCX, TXT, JSON, Markdown) as a searchable knowledge base |
| **Vector database** | LanceDB (embedded), ChromaDB, Pinecone, Weaviate, Qdrant and others |
| **MCP client** | Full MCP support via SSE – direct connection to mjEdit |
| **Workspace concept** | Isolated workspaces with their own documents, models and agents |
| **Agent system** | AI agents with tool access, web search and file operations |
| **Self-hosted** | Local, Docker or own server – no cloud dependency, full data sovereignty |

## RAG – your own compliance knowledge base

**Retrieval-Augmented Generation (RAG)** is the decisive advantage of AnythingLLM over pure chat assistants. Instead of letting the AI work only with general knowledge, you feed it with **your own compliance documents**:

**What you can embed:**

- BSI IT-Grundschutz compendium (PDF, 800+ pages)
- NIST SP 800-53 control catalog
- Your existing ISMS policies and procedures
- Audit reports from previous years
- Internal security policies and operating manuals
- OSCAL documents (JSON) from mjEdit
- Technical documentation of your IT infrastructure

**How RAG works:**

1. **Embedding:** You load your documents into an AnythingLLM workspace. The texts are decomposed into semantic vectors and stored in a local vector database.
2. **Retrieval:** When you ask a question, AnythingLLM searches for the most relevant text passages from your documents.
3. **Generation:** The LLM answers your question based on the found passages – with **source references** to the original documents.

**Result:** The AI knows your organisation, your policies and your infrastructure – and can incorporate this information directly into mjEdit workflows.

## MCP integration: AnythingLLM + mjEdit

The connection between AnythingLLM and mjEdit is made via the **SSE transport** of the MCP server:

**Configuration in AnythingLLM:**

```
{
  "mcpServers": {
    "mjEdit": {
      "url": "http://localhost:8765/sse",
      "transport": "sse"
    }
  }
}
```

After connection, all **88 MCP tools** of mjEdit are available to the AnythingLLM agent – combined with the

RAG knowledge from your own documents.

| Component | Contribution |
|---|---|
| **AnythingLLM RAG** | Knows your policies, standards, audit reports and infrastructure |
| **AnythingLLM agent** | Interprets requests, plans steps, calls tools |
| **mjEdit MCP tools** | Creates, validates and edits OSCAL documents |
| **mjEdit GUI** | Displays results in specialised tabs |

> **Note:** The following MCP use cases have only been tested in outline and represent a target scenario. Due to active development of the MCP API, the use cases may still change or deviate significantly from the workflow. The use cases merely illustrate the potential of AI in active interaction with mjEdit.

## Use case 1 – RAG-supported SSP creation from existing documentation

**Scenario:** Your organisation already has extensive security documentation in Word and PDF – but no OSCAL documents yet. You want to create a System Security Plan (SSP) based on your existing policies.

**What happens:**

1. AnythingLLM uses RAG to search your documents for relevant information (IP addresses, installed software, responsible persons, security measures)
2. The agent calls `oscal_generate_ssp_from_profile` and populates the system characteristics with the found data
3. Components and inventory items are extracted from your technical documentation
4. Control implementations are pre-populated with text passages from your policies
5. mjEdit opens the finished SSP in the specialised SSP tab for review

**Result:** An SSP that is not generic but reflects your actual infrastructure and real security measures – with traceable source references.

## Use case 2 – Intelligent audit preparation with knowledge base

**Scenario:** A BSI audit is approaching. You need to demonstrate that all relevant BSI Grundschutz modules are implemented. Previous audit reports and remediation documentation exist in various formats.

**What happens:**

1. RAG finds the relevant passages from the last audit report (findings, recommendations, deadlines)
2. Agent calls `oscal_list_controls` to compare the current SSP status
3. `oscal_generate_ap_from_ssp` creates a focused assessment plan for the identified controls
4. `oscal_generate_poam_from_ssp` generates a POA&M with concrete actions, deadlines and owners – extracted from the audit report
5. The documents are opened in mjEdit where you can edit the details

**Result:** Instead of weeks of manual work, a consistent audit preparation set is created in minutes – based on your real data.

### Use case 3 – Compliance chatbot for the entire team

**Scenario:** Your ISMS team has 12 members who regularly have questions about policies, controls and responsibilities. So far they have to search in various documents or ask the ISO.

**Typical team questions:**

> "Which controls from BSI Grundschutz affect our database servers?"

→ RAG searches the catalog and your SSP, lists relevant modules with implementation status.

> "Who is responsible for implementing OPS.1.1.3 and by when must the measure be completed?"

→ RAG finds the assignment in the SSP and the deadline in the POA&M; the agent calls `oscal_get_control` and `oscal_get_metadata` to deliver current data from mjEdit.

**Result:** An internal compliance knowledge portal that answers questions in seconds – with source references to the original documents and direct access to current OSCAL data in mjEdit.

## Comparison: AnythingLLM vs. cloud AI agents

| Criterion | Cloud agents (Claude, Copilot) | AnythingLLM (self-hosted) |
|---|---|---|
| **Data sovereignty** | Data leaves the organisation | Fully local / on-premises |
| **RAG knowledge base** | Limited (context window) | Unlimited (vector database) |
| **Own documents** | Upload per conversation | Persistently embedded |
| **MCP transport** | STDIO | SSE (HTTP) |
| **Cost** | API fees per token | One-time (hardware) + optional API |
| **Offline capability** | No (internet required) | Yes (with local models via Ollama) |
| **Multi-user** | Single-user context | Team workspaces with permissions |
| **Model choice** | Provider-bound | Freely selectable (OpenAI, Anthropic, local) |

# 5. The tabs in detail – all views, functions and use cases

mjEdit organises work in a multi-tab system. Each tab is specialised for a particular area and provides its own tools, context menus and keyboard shortcuts. The tabs work seamlessly together: changes in the text tab are reflected in the form tab, OSCAL tabs generate Markdown reports, and the PDF tab displays exported documents directly.

## 5.1 TEXT TAB (tab 0) – the JSON editor

The text tab is the heart of mjEdit: a fully-featured JSON editor with syntax highlighting, real-time validation, bracket matching and intelligent auto-repair.

### Feature overview

| Feature | Detail |
|---|---|
| **Syntax highlighting** | Colour coding for keys (purple), strings (green), numbers (blue), brackets (orange) |
| **Real-time validation** | JSON syntax checked on every change, error position (line/column) in the status bar |
| **Bracket matching** | Matching `{}` and `[]` are highlighted when the cursor is placed |
| **Auto-repair** | Missing commas, unclosed strings and invalid escape sequences corrected automatically |
| **Spell checking** | Wavy underline under misspelt words, customisable dictionary |
| **Line numbers** | Clickable line numbers with navigation |
| **Code folding** | Collapse / expand nested JSON structures |
| **Auto-indentation** | Intelligent indentation for nested structures |
| **Undo / redo** | Unlimited history for all changes |
| **Find & replace** | Regex support, live highlighting of all matches, forward / backward navigation |
| **Font persistence** | Chosen font and size saved between sessions |
| **Encoding detection** | Automatic detection of UTF-8, Latin-1, BOM markers |
| **JSON formatting** | Pretty-print with Ctrl+Alt+J, configurable indentation |
| **Schema validation** | Against OSCAL v1.2.1 or custom JSON schemas |

### Keyboard shortcuts in the text tab

| Shortcut | Action |
|---|---|
| Ctrl+Alt+J | Format JSON (pretty-print) |
| Ctrl+F | Find |
| Ctrl+H | Find & replace |
| F3 / Shift+F3 | Next / previous match |
| Ctrl+Shift+H | Replace all |
| Ctrl+Z / Ctrl+Y | Undo / redo |
| Ctrl+A | Select all |
| Ctrl+< | Insert snippet |

---

## 5.2 FORM TAB (tab 1) – qFORM form system & QC scripts

The form tab automatically converts JSON data into editable forms. The proprietary mjEdit qFORM format makes it possible to combine structured data entry with embedded Python scripts (QC scripts) – ideal for audit protocols, checklists, dynamic reports or structured data capture.

### Field types – automatic widget detection

| Field type | Widget | JSON detection | GUI element |
|---|---|---|---|
| Nested object | QGroupBox | Value is `{}` (dict) | Field group with frame |
| Array / list | QGroupBox (orange) | Value is `[]` (array) | List + "New item" button |
| QC script | QTextEdit (readonly) | Value starts with `"QC"`, >10 chars | Purple field, F12 to execute |
| Checkbox | QCheckBox | Value is `true` or `false` | Clickable checkbox |
| Dropdown | QComboBox | Value starts with `\|`, ≥2 pipes | Drop-down list |
| File path | QLineEdit + 🗀 | Value contains path pattern | Text field + browse button |
| Date | QLineEdit + DATE | Key contains date keyword or date format | Text field + calendar popup |
| Text (default) | AutoResizeTextEdit | Everything else | Auto-growing text field |

### Keyboard shortcuts in the form tab

| Shortcut | Action |
|---|---|
| F11 | Replace placeholders with values from other fields |
| F12 | Execute QC script in the current field |
| Tab / Shift+Tab | Navigate between form fields |
| Ctrl+S | Save form (synchronises back to JSON) |

## 5.3 MARKDOWN TAB (tab 2) – documentation and reports

The Markdown tab offers a full-featured Markdown editor with live preview, extended task list system and direct integration into OSCAL documentation.

### Keyboard shortcuts in the Markdown tab

| Shortcut | Action |
|---|---|
| Ctrl+B | **Bold** |
| Ctrl+I | *Italic* |
| Ctrl+Shift+D | Code block (‘‘‘) |
| Ctrl+Shift+X | ~~Strikethrough~~ |
| Ctrl+Shift+U | Unordered list |
| Ctrl+Shift+L | Ordered list |
| Ctrl+Shift+I | Insert image |
| Ctrl+Shift+K | Insert link |
| Ctrl+H | Find & replace |
| Ctrl+S | Save |

## 5.4 PDF TAB (tab 3) – viewer, annotator and redaction tool

The PDF tab displays PDF documents directly in mjEdit – without an external application. It supports multi-tab display, annotations, text search and secure redaction of sensitive content.

### Keyboard shortcuts in the PDF tab

| Shortcut | Action |
|---|---|
| Ctrl+F | Search in PDF |
| Ctrl+P | Print PDF |
| Ctrl+S | Save PDF with annotations |
| + / - | Zoom in / zoom out |
| Page Up / Down | Next / previous page |
| Alt+Q | Close PDF tab |

---

## 5.5 BROWSER-TAB (Plugin) – Integrated Web Browser

The Browser Tab integrates a fully functional Chromium-based web browser directly into mjEdit. This eliminates the need to switch contexts when looking up OSCAL references, NIST documentation, or BSI IT-Grundschutz Compendiums.

### Function Overview

| Function | Detail |
|---|---|
| **Chromium Engine** | Full web browser based on QWebEngineView |
| **Lazy Loading** | Browser engine is only loaded on first use – saves 200+ MB RAM at startup |
| **URL Navigation** | Address bar with input and Enter navigation |
| **Forward/Back/Reload/Home** | Standard browser navigation |
| **Multiple Browser Tabs** | Open additional sub-tabs within the browser tab |
| **HTTPS Status** | Green/red padlock icon indicates connection security |
| **Zoom Control** | 75% – 300% zoom |
| **Bookmarks** | Two-tier system: global (team) + private (personal) bookmarks with tag-based organisation |
| **Configurable Homepage** | Default homepage adjustable in config |
| **Cookie Consent** | Automatic acceptance (configurable) |
| **External URL Integration** | Links from Markdown documents automatically open in the browser tab |
| **HTTPS Warning** | Warning for insecure HTTP connections |

## GPU Stability Optimisations

The browser tab uses specific flags for maximum stability:

- Disabled GPU sandbox for compatibility with virtualised environments
- Adjusted rendering for terminal servers and remote desktops

## Keyboard shortcuts in the browser tab

| Shortcut | Action |
|---|---|
| Ctrl+Tab | Next browser sub-tab |
| Ctrl+Shift+Tab | Previous browser sub-tab |
| Ctrl+W | Close browser sub-tab |

## Bookmark System: Global and Private Bookmarks

The browser tab features a **two-tier bookmark system** that distinguishes between team-wide and personal bookmarks – a concept not found in standard browsers.

**Global bookmarks** are intended for the entire team: a shared collection of compliance references, OSCAL documentation, and audit resources accessible to all users. The file `global-browser-bookmarks.json` is located in the `data/` directory and can be shared within the team via version control (Git).

**Private bookmarks** are personal bookmarks for individual users. They are stored in `config/browser-private-bookmarks.jso` and are automatically tagged as `private`. Using the "Quick Bookmark" feature, they can be created with a single click – no dialog, no input required.

| Property | Global Bookmarks | Private Bookmarks |
|---|---|---|
| **Purpose** | Team resources, shared references | Personal collection |
| **Location** | `data/global-browser-bookmarks/` | `config/` |
| **Adding** | Dialog with title, notes, tags, shortcut | One click – no dialog needed |
| **Typical Content** | BSI compendium, NIST reference, ISO standards | Personal research links, notes |
| **Team Sharing** | Shareable via Git | Local only |

**Adding bookmarks – three methods:**

All bookmark functions are accessible via the context menu (right-click on the webpage or the URL bar):

| Context Menu Option | Function |
|---|---|
| ⭐ **Add to Global Bookmarks** | Opens a dialog with a pre-filled title, URL, notes field, tag input, and user shortcut |
| 📌 **Quick Bookmark** | Saves the current page **instantly** as a private bookmark – tags are automatically extracted from the page title |
| 🗂 **Open Bookmark Manager** | Opens the bookmark manager with all saved bookmarks |

**Tag-based organisation:**

Instead of rigid folder hierarchies, mjEdit uses a flexible **tag system**. Each bookmark can have multiple tags – displayed as coloured chips with a delete button. The advantages:

- A bookmark can belong to multiple categories simultaneously (e.g., `oscal`, `nist`, `ssp`)
- Tags are automatically generated from the page title when using Quick Bookmark (stop words in German and English are filtered, minimum 4 characters, maximum 5 tags)
- In the bookmark manager, the live search filters by title, tags, notes, creator, and date

**Bookmark Manager:**

The Bookmark Manager (🗂 "Open Bookmark Manager") displays all global and private bookmarks in a table:

| Column | Content |
|---|---|
| **Title** | Page title (double-click opens the URL in the browser) |
| **Notes** | Optional description of the bookmark |
| **Tags** | Coloured tag chips for categorisation |
| **Created** | Date, time, and user shortcut |
| **Actions** | ✏ Edit, 🗑 Delete |

A live filter input above the table searches all fields in real time – ideal for quickly finding the right reference in a growing collection.

**Practical Example:**

An ISB team is collaboratively working on OSCAL documentation. The global bookmarks include:

- BSI IT-Grundschutz Compendium (Tags: `bsi`, `grundschutz`)
- NIST OSCAL Reference (Tags: `nist`, `oscal`, `reference`)
- NIST SP 800-53 Controls (Tags: `nist`, `controls`, `800-53`)
- ISO 27001:2022 Annex A (Tags: `iso`, `27001`, `annex-a`)

Each team member also has private bookmarks for their own research – e.g., CVE entries, vendor documenta-

tion, or internal wiki pages. The private collection remains on their own computer, while the global collection is synchronised via Git.

## Use Cases

**Use Case 1 – Look up OSCAL reference:** While editing an SSP, you need to check the exact structure of an OSCAL element. Instead of switching to the browser, you open the NIST OSCAL reference (pages.nist.gov/OSCAL-Reference/) directly in the browser tab. The bookmark feature saves frequently used reference pages.

**Use Case 2 – View BSI IT-Grundschutz Compendium:** You are implementing BSI controls in a catalogue and need the original texts from the BSI Compendium. The browser tab displays the BSI website alongside your work – no context switching, no window switching.

**Use Case 3 – Training materials during work:** New employees can read OSCAL tutorials and documentation in the browser tab while simultaneously creating their first OSCAL documents in other tabs – all within one application.

**Use Case 4 – Open Markdown links directly:** Your Markdown documentation contains links to external resources (e.g. CVE database, NIST NVD). Clicking on the link automatically opens the page in the browser tab instead of launching an external program.

## Save Websites as OSCAL Back-Matter Resource

A unique feature of the browser tab is its direct integration with the OSCAL system: any displayed webpage can be added as a **Back-Matter Resource** to an open OSCAL document via right-click. OSCAL uses Back-Matter as a standardised appendix for references, evidence, and external sources – and mjEdit makes creating these references as simple as a right-click.

**Two options in the context menu:**

| Menu Item | Description | Prerequisite |
|---|---|---|
| 📎 **Create as OSCAL Attachment** | Creates a Back-Matter Resource with URL reference, title, and description | OSCAL plugin is active |
| 🔗 **Create New OSCAL Resource** | Advanced option with remarks, media type, and source URL | At least one OSCAL tab is open |

**How it works:**

1. **Open the webpage** – Navigate to the desired source in the browser tab (e.g., BSI IT-Grundschutz Compendium, CVE database, NIST reference)
2. **Right-click → Create Resource** – The dialog opens with the pre-filled URL and page title
3. **Add title and description** – Optionally, include remarks on why this source is relevant

4. **Select OSCAL tab** – If multiple OSCAL documents are open, choose the target document
5. **Insert** – The resource is automatically added to the `back-matter.resources` list of the OSCAL document

**What is saved in the OSCAL file:**

```
{
  "uuid": "auto-generated-uuid-v4",
  "title": "BSI IT-Grundschutz Compendium 2024",
  "description": "Reference for SYS.1.1 control",
  "props": [
    {"name": "created-by", "value": "mjEdit"},
    {"name": "created-at", "value": "2026-04-19T14:30:00+02:00"},
    {"name": "source-url", "value": "https://www.bsi.bund.de/..."}
  ],
  "rlinks": [
    {"href": "https://www.bsi.bund.de/...", "media-type": "text/html"}
  ],
  "remarks": "Basis for implementing control SYS.1.1"
}
```

**Supported OSCAL document types:** The insertion works in **all 7 OSCAL tabs** – Catalog, Profile, SSP, Assessment Plan, Assessment Results, POAM, and Component Definition. The insertion logic automatically detects whether the tab uses a Pydantic model (Catalog, Profile) or a dictionary-based data model (SSP, AP, AR, POAM, Component) and selects the appropriate strategy.

**Use cases:**

- **Audit trail:** During an audit, save links to CVE entries, BSI alerts, or vendor documentation directly in the Assessment Results document
- **Compliance evidence:** Link policies, certificates, or compliance reports as Back-Matter Resources in the SSP
- **Catalog reference:** When creating a company-specific catalog, link the original sources (NIST, BSI, ISO) directly in the document

**Added value:** The integrated browser eliminates the context switch between editor and web browser. Especially when working with OSCAL specifications, BSI compendiums, or NIST references, the direct integration saves significant time and reduces errors caused by copy-pasting between applications.

---

## 5.6 OSCAL-TABS (Plugin) – Specialised Editors for Compliance Documents

The OSCAL tabs are dynamic, specialised editors that provide a customised interface for each of the 8 OSCAL document types. They open automatically when mjEdit detects an OSCAL file.

### Common Features of All OSCAL Tabs

| Function | Detail |
|---|---|
| **Automatic Detection** | OSCAL files are recognised upon opening and loaded into the appropriate tab |
| **Tree Structure (TreeView)** | Hierarchical navigation through all document elements |
| **Context Menus** | Specialised actions per document type via right-click |
| **Schema Validation** | Real-time validation against OSCAL v1.2.1 |
| **UUID Auto-Generation** | RFC 4122-compliant UUIDs for new elements |
| **Metadata Editor** | Dialog for title, version, last-modified, roles, parties |
| **Markdown Export** | Each OSCAL tab can be exported as a Markdown report |
| **Tab Registry** | Central management of all open OSCAL tabs |
| **Bidirectional Synchronisation** | Changes in the tab update the JSON data and vice versa |

### 6.6.1 Catalogue Tab

Editing of control frameworks (e.g. BSI IT-Grundschutz, NIST SP 800-53).

| Function | Detail |
|---|---|
| Group Navigation | Hierarchical tree structure with groups and controls |
| Control Details | Title, description, properties, parts, parameters |
| Control Search | Filter by ID, title, or free text |
| Cascade Selection | Select entire groups with a single click |
| Nested Groups | Up to 3 levels deep (Group → Subgroup → Control) |
| Import Controls | Adopt from other catalogues |
| Back Matter | Attach Base64-encoded resources (PDFs, images) |

**Use Case:** You create a company-specific catalogue by selecting relevant controls from NIST SP 800-53 and BSI IT-Grundschutz, organising them into groups, and adding your own descriptions.

### 6.6.2 Profile Tab

Creation of baselines by selecting and tailoring controls from catalogues.

| Function | Detail |
|---|---|
| Baseline Selection | Select controls from imported catalogues |
| Profile Tailoring | Add, restrict, or replace controls |
| Parameter Tuning | Set organisation-specific values (e.g. password length) |
| Profile Resolution | Generate a flat catalogue from profile + base catalogue |
| Profile Merge | Combine multiple profiles into one |
| Exclusion Lists | Explicitly exclude irrelevant controls |
| Save As | Dialog with target directory selection |

**Use Case:** You create a "Virtual Windows Server 2019" profile by selecting relevant controls from the NIST catalogue, adjusting parameters such as the audit retention period, and excluding non-applicable controls (e.g. physical security).

### 6.6.3 SSP Tab (System Security Plan)

The most comprehensive OSCAL tab – documents a complete system with components, inventory, and control implementation.

| Function | Detail |
|---|---|
| System Definition | Metadata, authorisation boundary, system type |
| Component Registry | Software, hardware, services, policies as building blocks |
| Inventory Management | Specific assets with network properties (IP, MAC, FQDN) |
| Component-Inventory Linkage | OOP model: Component = class, Inventory = instance |
| Control Implementation | By-component assignment: Which component fulfils which control |
| Roles & Parties | Assign responsibilities |
| Network Discovery | Automatically add IP addresses, hostnames to inventory |
| CSV Import/Export | Import inventory from Excel/CMDB |
| Network Diagram | Automatic NWDiag generation from inventory data |
| Compliance Check | Analyse control coverage (implemented/partial/planned) |
| Impact Analysis | Determine affected components via CVE |
| Dynamic Documentation | Auto-update on inventory changes |

**Use Case:** You document your server landscape: Define "Apache HTTP Server" and "MySQL" as components (classes), create inventory items for each specific server (instances), link both, and document per component which controls are implemented.

### 6.6.4 Assessment Plan Tab

Planning of security assessments and audits.

| Function | Detail |
|---|---|
| Assessment Activities | Define testing methods (Review, Interview, Test, Examine) |
| Assessment Subjects | Select systems/components to be assessed |
| Scope Definition | Define boundaries and timeframe of the assessment |
| Assessment Team | Assign assessors and roles |
| Generate from SSP | Automatically derive the assessment plan from the existing SSP |

**Use Case:** You are preparing for an ISO 27001 audit: The assessment plan is generated from the SSP, including all implemented controls, testing methods (e.g., interview with admin, testing firewall rules), and the schedule.

### 6.6.5 Assessment Results Tab

Documentation of assessment results and findings.

| Function | Detail |
|---|---|
| Record findings | Weaknesses with severity and description |
| Observations | Additional insights and notes |
| Evidence linking | Links to supporting documents and resources |
| Risk assessment | Severity classification (critical, high, medium, low) |
| Assessment log | Audit trail of all assessment steps |
| POA&M linking | Transfer findings directly into the action plan |
| Generate from assessment plan | Results template from an existing plan |

**Use Case:** During the audit, document your findings: "Firewall rules not documented" (Severity: high), "Backup test not performed" (Severity: medium). Each finding is directly transferred as a POA&M item into the action plan.

### 6.6.6 POA&M Tab (Plan of Action & Milestones)

Tracking of actions and remediation after audits.

| Function | Detail |
|---|---|
| Remediation Items | Actions with target date and description |
| Risk Prioritisation | Impact-based prioritisation |
| Milestone Tracking | Start → Planned → Completion with progress control |
| Status Management | Planned, In Progress, Completed, Cancelled |

| Function | Detail |
|---|---|
| Resource Allocation | Responsible persons and budget per action |
| Automatic Status Updates | When linked with assessment results |
| Generate from SSP/AR | POA&M directly from existing documents |

**Use Case:** After the audit, you automatically create an action plan from the assessment results: 5 critical actions (deadline: 30 days), 12 medium (90 days), 3 low (180 days). Each action has a responsible person and milestones.

### 6.6.7 Component Definition Tab

Library of reusable system components.

| Function | Detail |
|---|---|
| Component Library | Reusable building blocks (software, hardware, service, policy) |
| Control Implementation | Which controls are implemented by each component |
| Implementation Statements | Description of the specific implementation |
| Revision History | Remarks history of versions |
| Import/Export | Exchange components between projects |

**Use Case:** Your organisation uses standardised Linux servers. You define an "Ubuntu 24.04 LTS" component with all implemented controls (patch management, logging, access control). This component is reused in every new SSP.

### 6.6.8 Mapping Collection Tab — The Offline-Supported AI Mapping Cockpit

The **OSCAL Mapping Tab** is one of the standout unique features of mjEdit. It allows controls from two different OSCAL documents (source → target) to be systematically mapped to each other — between catalogues (e.g. NIST SP 800-53 ↔ BSI IT-Grundschutz), between a catalogue and a profile, or between two profiles with different tailoring levels. The result is an OSCAL-compliant **Mapping Collection Document** (`mapping-collection`), which records each mapping with source/target UUID, relation type (`equivalent-to`, `subset-of`, `superset-of`, `intersects-with`), confidence score, and optional notes.

> **Screenshot Note:** *Mapping Tab in split layout: on the left, the source control list with search/filter bar; on the right, the target control list; in the middle, the suggestion list with confidence bars (0–100%) and buttons "Accept", "Reject", "Auto-Accept".*

**Feature Overview:**

| Feature | Detail |
|---|---|
| **Auto-Suggest with Three Methods** | `syntactic` (token overlap), `semantic` (sentence transformer embeddings), `auto` (combination + score fusion) |
| **Offline AI Matching** | Local sentence transformer model (`paraphrase-multilingual-MiniLM-L12-v2`, 3.9 GB), no cloud, no data sharing |
| **Multilingual** | Controls in DE/EN/FR/IT can be directly compared — embeddings are multilingual |
| **Auto-Accept Threshold** | Configurable confidence threshold (default 0.4); suggestions above this are automatically accepted |
| **Bidirectional & Visual** | Linked controls are colour-coded; unmapped controls are immediately visible |
| **Mapping Status** | `complete`, `not-complete`, `draft`, `deprecated`, `superseded` per mapping (OSCAL 1.2.1 schema enum) |
| **Bulk Operations** | Accept all suggestions above the threshold, reject all, filter by score range |
| **Schema Validation** | Validation against the OSCAL `mapping-collection` schema before every save |
| **Markdown Export** | Complete mapping table as a report (with confidence column and justifications) |

> **Screenshot Note:** *Detail view of an expanded suggestion: source control "A.5.1 Information security policies" (ISO 27001) next to target control "AC-1 Policy and Procedures" (NIST SP 800-53) with confidence 0.87, justification of the three methods (syntactic 0.42, semantic 0.91, combined 0.87), and a free notes field.*

**Offline-Supported AI Matching in Detail** **Why offline?** Compliance documents — especially SSPs and internal control catalogues — often contain highly sensitive information about system architectures, vulnerabilities, and security measures. They must not leave the organisation. Cloud-based AI mappers (OpenAI, Anthropic…) are often simply **not permissible** here due to legal (GDPR, BSI C5, BSI Minimum Standards, ITSiG, EU AI Act) and contractual reasons. Therefore, mjEdit uses a **purely local** multilingual sentence-transformer model, which is downloaded once and stored in the user profile (`%APPDATA%\mjEdit\models\`). **No token leaves the computer.**

**How does it work?** The mapping service maps the title and description of each control to a 384-dimensional embedding vector and calculates the cosine similarity of all source-×-target pairs. In `auto` mode, semantic and syntactic similarity (token overlap, lemmatisation, Levenshtein) are combined with weighted importance. The model is trained in 50+ languages — a BSI Basic Protection module in German is reliably matched to the

English-language NIST control without the user needing to translate the texts beforehand.

**Performance:** On a typical office notebook (CPU-only), approximately 2,000–10,000 comparison pairs are processed per second. A complete auto-suggest analysis between 200 BSI modules and 800 NIST controls (=160,000 pairs) typically takes less than 30 seconds.

> **Screenshot Note:** *Status bar during auto-suggest with progress bar "768 of 2,304 comparison operations ... ETA 12 s" and log file entry* `[MAPPING-USE] Using model 'paraphrase-multilingual-MiniLM-L12-v2' for mapping: nist-800-53.json -> bsi-grundsc`

**Three use cases where the mapping tab truly shows its value** Use Case 1 — Multi-Framework Compliance: BSI IT-Grundschutz ↔ NIST SP 800-53 ↔ ISO 27001

**Initial situation:** A medium-sized international company needs to demonstrate BSI IT-Grundschutz compliance to German authorities, NIST SP 800-53 compliance to US customers, and ISO 27001 compliance to the corporate headquarters. A manual cross-reference table in Excel is error-prone and constantly outdated.

**With the Mapping Tab:** 1. Open three catalogues (BSI, NIST, ISO). 2. Create a mapping collection document for each pair (BSI↔NIST, BSI↔ISO, NIST↔ISO). 3. Trigger Auto-Suggest with the `auto` method — the system suggests all plausible correspondences within 1–2 minutes. 4. Automatically accept the highly confident 60–70% of mappings using an auto-accept threshold of 0.7. 5. Manually review and either accept or reject the remaining suggestions. 6. Export as Markdown to generate a cross-reference matrix for the audit.

**Added value:** Instead of 4–6 person-days of manual research, only half a day is needed. The mapping is versionable (each mapping file can be under version control), reusable, and audit-proof.

> **Screenshot note:** *Three tabs opened side by side (BSI↔NIST, BSI↔ISO, NIST↔ISO), each with a status bar and the number of mapped controls; in the background, the cross-reference table displayed as a Markdown preview.*

Use Case 2 — Trace and Visualise Profile Tailoring

**Initial situation:** A **high-baseline profile** is derived from the NIST SP 800-53 catalogue, which is then tailored into an even stricter organisation-specific profile. Auditors want to trace the tailoring path: Which controls were adopted, which were modified, and which were removed?

**With the Mapping Tab:** The tab works not only between catalogues but also between a catalogue and a derived profile. It immediately shows:

- **Adopted controls** (semantic = 1.0, syntactic = 1.0): unchanged.
- **Modified controls** (semantic > 0.85, syntactic < 0.95): content adjusted, e.g., parameter tuning.
- **Removed controls** (in the source catalogue but no mapping entry): excluded from the baseline.
- **New organisation-specific controls** (in the profile but no source mapping): custom additions.

**Added value:** The tab automatically generates the justification documentation that many auditors strictly

require (“Why was control XY excluded?”). Mapping notes are directly linked to audit evidence.

> **Screenshot note:** *Filter view “Show only removed controls”: 12 BSI modules that were omitted in the tailored profile, each with a notes field for justification.*

Use Case 3 — Gap Analysis and Risk Identification in Framework Migration

**Initial Situation:** A federal authority is migrating from an existing BSI IT Baseline Protection 2023 framework to the new **BSI IT Baseline Protection++** (successor from 2026/2027). The central question: **Which existing modules and requirements have no equivalent in the new framework?** These are potential risk sources.

**Using the Mapping Tab:** 1. Select BSI IT Baseline Protection 2023 as the source and BSI IT Baseline Protection++ as the target. 2. Use Auto-Suggest with a low threshold (0.3) to make weak correlations visible. 3. Apply the “Unmapped Source Controls” filter — these are modules without an equivalent in the new framework. 4. For each unmapped entry: Perform a risk analysis to determine whether the requirement is still necessary (→ as an organisation-specific addition in the profile) or is covered differently in the new framework. 5. Export to Markdown as a **Gap Analysis Report** with a risk score column.

**Added Value:** A task that would take weeks without tooling (manually reviewing 800+ modules/requirements) is reduced to a few hours. The gap list forms the basis for a well-informed migration decision from Baseline Protection 2023 to Baseline Protection++.

> **Screenshot Note:** *Filter mode “Unmapped Source Controls” with red highlighting; the right sidebar shows the statistic “23 out of 312 controls unmapped (7.4%)” and a “Create Gap Report” button.*

**From Mapping to the OSCAL Document Chain — Generating Additional Documents** The `mapping-collection` document created in the Mapping tab is not the final result — it is the **foundation** for the automatic generation of additional OSCAL documents. From a completed mapping, the following derived documents can be generated via the **“Generate OSCAL Documents…”** button in the Mapping tab:

| Generated Document | Purpose |
|---|---|
| **Resolved Profile** (`profile`) | A tailoring profile from the target catalogue that contains exactly the controls accepted in the mapping as a baseline. *Purpose:* Direct use as a compliance requirement for auditors. |

| Generated Document | Purpose |
|---|---|
| **SSP Proposal** (`system-security-plan`) | An SSP stub with `control-implementation` entries for each mapped target control, prefilled with `description` from the source control. *Purpose:* Dramatically accelerates SSP creation — the organisation-specific descriptions from the source framework are adopted 1:1 and only need refinement. |
| **Cross-Reference Report** (Markdown) | Table of all mappings with confidence, justification, and status. *Purpose:* Audit evidence, management reporting, internal training material. |
| **Gap Analysis Report** (Markdown) | List of all unmapped source or target controls with justification fields. *Purpose:* Risk identification during framework migrations, basis for POA&M entries. |
| **Component Definition Stub** (`component-definition`) | For each component, a list of the mapped target controls with adopted implementation statements. *Purpose:* Building reusable component libraries (e.g., "Our standard Linux server hardening against NIST and BSI"). |
| **POA&M Stub** (`plan-of-action-and-milestones`) | For each unmapped source control, an entry is automatically created with the status "open". *Purpose:* Systematically track the closure of control gaps in the new framework. |

> **Screenshot Note:** *"Generate OSCAL Documents" dialog with a checkbox list of the six options, target directory selection, and preview area showing the planned output set.*

**Where the Mapping Tab Brings Automation Benefits**

| Task | Manual | With Mapping Tab | Time Savings |
|---|---|---|---|
| Cross-referencing NIST↔BSI (200×800 controls) | 4–6 person-days | 2–3 hours | **> 95 %** |
| Tailoring documentation (catalogue→profile) | 1–2 person-days | 30 minutes | ~ 90 % |
| Pre-filling SSP with implementation texts | Weeks (manual copy-paste) | Minutes (generated from mapping) | **> 95 %** |

| Task | Manual | With Mapping Tab | Time Savings |
|---|---|---|---|
| Gap analysis during framework migration | 2–3 weeks | 1 day | ~ 90 % |
| Audit cross-reference report | 2 days | 1 click (Markdown export) | **> 99 %** |
| POA&M creation from gaps | Days | Minutes | **> 95 %** |

Especially valuable: **Reusability**. A BSI↔NIST mapping created once can be used by all projects within the organisation, automatically updated during framework updates (auto-suggest for the new version, diff view of changes), and version-controlled in Git — as a valuable core asset of the compliance organisation.

> **Screenshot Note:** *Mapping file in the file tree view with version history ("v1.0 BSI Grundschutz 2023↔NIST rev5", "v2.0 BSI Grundschutz++↔NIST rev5") and diff display of modified mappings.*

---

## 5.7 Cross-Functional Features

### File Management

| Function | Detail |
|---|---|
| Open/Save | JSON, Markdown, text with encoding detection |
| Export | XML, CSV, PDF, HTML |
| Templates | Create files from templates (with variable substitution) |
| Snippets | Insert code/JSON fragments using Ctrl+< |
| Auto-Open | Restore the last file on startup |
| Project Directories | Quick access to project folders |

---

## 5.8 File Tree View – Central Project Management

The **File Tree View** is much more than a simple folder overview. It serves as a **central cockpit** for managing flexible project directory structures and is the primary hub for all file-oriented actions in mjEdit.

## Building a Flexible Project Structure

OSCAL compliance projects typically consist of a variety of interconnected documents – catalogues, profiles, SSPs, assessment plans, assessment results, and POA&M files. mjEdit recommends and supports a clear, navigable directory hierarchy:

```
010-ISMS/                          ← Project directory (via Config: sys_project_path)
├── 010-Catalogues/
│   ├── bsi-grundschutz-2024.json  ← OSCAL catalogue
│   └── nist-sp-800-53.json
├── 020-Profiles/
│   ├── baseline-server.json       ← OSCAL profile (references catalogue)
│   └── baseline-client.json
├── 030-SSP/
│   ├── ssp-webserver-prod.json    ← System Security Plan (references profile)
│   └── ssp-db-cluster.json
├── 040-Assessment/
│   ├── ap-2026-q1.json            ← Assessment Plan
│   └── ar-2026-q1.json            ← Assessment Results
├── 050-POAM/
│   └── poam-2026.json             ← Plan of Action & Milestones
└── 060-Reports/
    ├── compliance-report-q1.md    ← Markdown report
    └── audit-evidence.pdf
```

**Configuring the project directory:** Use *Settings → Project Path* (or `sys_project_path` in the config) to define the root directory. This directory appears as the root in the tree view and also serves as the base for resolving relative paths between OSCAL documents.

## Functions at a Glance

| Function | Detail |
|---|---|
| **Lazy Loading** | Large directories (1,000+ files) are loaded incrementally |
| **Coloured File Icons** | Differentiate file types by colour – configurable per extension |
| **Real-Time Filter** | Case-insensitive instant search filters the entire tree, including subfolders |
| **Context Menu** | Right-click opens context-specific actions based on file type |
| **Template-Based Creation** | Create new files directly from a template in the desired directory |
| **Backup Indicator** | Icon shows whether backups are available for a file |
| **Multi-Selection** | Select multiple files simultaneously for batch operations |

| Function | Detail |
|---|---|
| **Rename via F2** | Rename files or folders directly in the tree |
| **Drag & Drop** | Move files using drag and drop |
| **New Folder** | Create subfolders directly in the tree |

## Context Menu – Actions via Right-Click

The context menu displays different options depending on the file type:

| Menu Entry | File Type | Action |
|---|---|---|
| **Open** | All | Opens the file in the appropriate tab (OSCAL → OSCAL tab, .md → MD tab) |
| **Open as Text** | All | Always opens the file in the Text tab, bypassing OSCAL/MD detection |
| **Rename** | All | Rename file/folder (F2) |
| **Copy** | All | Copy file to clipboard |
| **Move** | All | Move file |
| **Delete** | All | Delete file (with confirmation dialog) |
| **New File from Template** | Folder | Open template selection dialog, create file in folder |
| **New Folder** | Folder | Create subfolder |
| **Create Backup** | JSON/Markdown | Create an instant backup of the file |
| **Show Versions** | JSON/Markdown | Open version archive dialog |

> **Tip: "Open as Text"** — OSCAL files are usually opened automatically in the specialised OSCAL tab. With " Open as Text," you can deliberately open the file in the Text tab to view the raw content or edit it manually without OSCAL routing interfering.

## Real-Time Filter – Quickly Navigate Large Projects

The filter above the tree view is a powerful navigation tool for large OSCAL projects with many files. The search is:

- **Case-insensitive** – "SSP", "ssp", and "Ssp" find the same files
- **Recursive** – searches all levels of the directory hierarchy
- **Instant** – filters with every input without delay
- **Path-searching** – searches in file names **and** directory names

**Examples:**

| Input | Found Files |
|---|---|
| `ssp` | `ssp-webserver.json`, `ssp-db.json`, `030-SSP/` |
| `q1` | `ap-2026-q1.json`, `ar-2026-q1.json` |
| `grundschutz` | `bsi-grundschutz-2024.json` |
| `.md` | All Markdown files in the project |

## Configure Coloured File Icons

You can customise file extensions with colours via *Settings → File Colours*:

| Default Configuration | Colour | Description |
|---|---|---|
| `.json` | Green | JSON and OSCAL files |
| `.md` | Blue | Markdown documentation |
| `.pdf` | Red | PDF documents |
| `.txt` | Grey | Text files |
| `.csv` | Orange | Tabular data |

The configuration is done in `config/config.json` under the key `file_color_map`. New extensions can be added at any time without restarting.

## Use Cases

**Use Case 1 – Structuring an OSCAL project directory:** You are starting a new ISMS project. In the tree view, you create the desired folder structure (Catalogues/, Profiles/, SSP/, etc.) via the context menu. Then, you use "New File from Template" for each OSCAL document type and immediately get pre-filled starter documents in the correct location.

**Use Case 2 – Exchanging files between projects:** You are working on two client projects simultaneously. Using drag and drop in the tree view, you move a proven SSP profile from Project A into the profile directory of Project B. mjEdit automatically recognises the OSCAL type and the updated relative path references the next time you open it.

**Use Case 3 – Quick navigation using filters:** Your project contains 200+ files. You are searching for all Assessment Results documents: simply type "ar-" into the filter field – only the relevant files remain visible. A double-click immediately opens the desired file.

**Use Case 4 – Checking a file as text:** A colleague has sent you an OSCAL profile file that cannot be opened. By right-clicking → "📄 Open as Text," you view the raw content in the text tab – you immediately notice that a required field is missing. After correcting it, the file opens correctly in the profile tab.

---

## 5.9 Backup and Versioning Concept

mjEdit provides a **two-tier safety net** for all edited files: automatic **backups** (time-based full copies) and a **version archive** (manual snapshots). Both mechanisms operate independently and complement each other perfectly.

### Level 1: Automatic Backup System

The automatic backup system creates compressed copies in the background – without manual intervention and without interrupting work.

**How it works:**

1. When **opening a file**, mjEdit checks if the last backup is older than the configured interval.
2. When **saving** (Ctrl+S), an automatic backup can optionally be created.
3. **Compressed storage** as a `.json.gz` archive – OSCAL files typically shrink to 5–10% of their original size.
4. **Naming convention:** `{originalfilename}_YYYY-MM-DD_HH-MM-SS.json.gz`
5. **Backup directory:** `config/backups/{projectpath-hash}/`

**Configurable parameters** (via *Settings → Backup*):

| Parameter | Default value | Description |
|---|---|---|
| **Automatic Backups** | enabled | Master switch for automatic backups |
| **Backup Interval** | 30 minutes | Minimum interval between two automatic backups |
| **Max. Backups per File** | 10 | Oldest backups are automatically deleted |
| **Compression** | enabled | GZIP compression saves up to 95% of storage space |
| **Backup on Save** | enabled | Additional backup on every Ctrl+S |

**Manual Backup (Ctrl+Shift+B):** Using the shortcut or the menu *File → Create Backup*, a compressed copy of the current file is created immediately. Ideal before major changes or before testing an automatic OSCAL transformation.

**Backup Status in the Tree View:** Files with existing backups are marked with a small icon in the tree view. This allows you to see at a glance which files are backed up – and which are not.

### Level 2: Version Archive

The version archive is an explicit collection of named snapshots. Unlike automatic backups, versions can be given a descriptive name – similar to commits in a version control system.

**Open the Version Archive Dialog:**

- Via *File → Show Versions*
- Via right-click in the tree view → "🕑 Show Versions"

- Via the context menu in the text tab

**Version Archive Dialog – Features:**

| Feature | Description |
|---|---|
| **Version List** | All versions with timestamps, file size, and description |
| **Preview** | Check the content of the selected version before restoring it (JSON-formatted) |
| **Show Diff** | Line-by-line comparison between the selected version and the current file |
| **Restore** | Selected version replaces the current file (after confirmation) |
| **Delete** | Remove version from the archive |
| **Export** | Save version as a separate JSON file in another location |

## Secure Recovery Workflow

The recovery process is deliberately designed in multiple steps to prevent accidental data loss:

```
1. Open the versions dialog
       ↓
2. Select a version from the list
       ↓
3. Check the preview → is it the correct version?
       ↓
4. Optional: Show diff with the current file
       ↓
5. Click "Restore"
       ↓
6. Confirmation dialog: "Really overwrite the current file?"
       ↓
7. Automatic backup of the current file BEFORE overwriting
       ↓
8. Version is restored and opened in mjEdit
```

**Safety net during recovery:** Before each recovery, a backup of the current file is automatically created. This also allows undoing a recovery – a two-step safety net.

| Scenario | Recommended Action |
|---|---|

### Backup and Versioning in Interaction

| Scenario | Recommended Action |
|---|---|
| Securing daily work | Keep automatic backups enabled (default setting) |
| Before a major editing step | Ctrl+Shift+B → Manual backup with timestamp |
| Documenting a milestone (e.g. "v1.0 Review") | Create a version with a descriptive name |
| File accidentally overwritten | Backup dialog → Select oldest version → Preview → Restore |
| Comparing two versions content-wise | Version archive → Show diff |
| Copying backup to another location | Export version → Save file in another directory |

**Practical Example – Securing OSCAL Profile Tailoring:**

1. You open a trusted BSI baseline protection profile and want to tailor it for a new client
2. Ctrl+Shift+B → Backup with the label "Original BSI Profile before Client Project X"
3. You begin tailoring – exclude 15 controls, adjust 8 parameters
4. While testing the profile resolution, you notice that a control was mistakenly excluded
5. Backup dialog → Select version from step 2 → Preview shows the original → Restore

### Technical Details

| Aspect | Details |
|---|---|
| **Compression Format** | GZIP (Python `gzip` module), Level 6 (balance of speed/ratio) |
| **Storage Location** | `config/backups/` (platform-independent path via `path_helper`) |
| **Max. Memory Usage** | Configurable; default: 10 backups × compressed size |
| **Thread Safety** | Backups are created asynchronously in the background (no UI freeze) |
| **File Integrity** | CRC32 check when reading compressed backups |
| **Portability** | Backups can be manually copied and opened on other systems |

---

## 5.10 OSCAL: Intelligent Path Correction for Unreachable Absolute Paths

OSCAL documents reference each other via **Href attributes** – profile files reference catalogues, SSPs reference profiles, and so on. If absolute paths are used (e.g., `C:\Users\max\Projects\catalogue.json`) and the project structure is later copied to another computer or moved to a different directory, these references become invalid.

mjEdit automatically detects this situation and offers an **intelligent path correction** with two resolution levels.

## The Problem: Absolute Paths After Moving

**Typical Scenario:**

1. OSCAL documents were created on Computer A: Profile references catalogue via absolute path `C:\Users\alice\ISMS\catalogue.json`
2. Project directory is copied to a USB stick or transferred to Computer B via Git
3. On Computer B, mjEdit opens the profile – the absolute path reference `C:\Users\alice\...` is inaccessible
4. **Previous Behaviour:** Simple warning message, no solution offered
5. **New Behaviour:** Intelligent analysis, preview of corrections, automatic conversion to relative paths

## Two-Step Path Resolution

When mjEdit detects unreachable absolute paths, it automatically checks whether the referenced files can be found locally:

**Step 1 – Source Directory Search:** Searches for the file name (e.g., `bsi-grundschutz.json`) recursively in the directory of the OSCAL source file and its subdirectories. If the file is found, mjEdit calculates the relative path from the location of the source file.

```
Absolute path (unreachable):      C:\Users\alice\ISMS\Catalogues\bsi-grundschutz.json
Source file located in:           /ProjectDirectory/Profiles/baseline.json
File found in:                    /ProjectDirectory/Catalogues/bsi-grundschutz.json
→ Suggested relative path:        ../Catalogues/bsi-grundschutz.json
```

**Step 2 – Project Index Search (project mode only):** If Step 1 yields no result, mjEdit builds an index of all files in the project directory (`{FileName → all absolute paths}`) and searches it for the file name of the unreachable reference.

```
File not found in source directory.
Project index contains: "bsi-grundschutz.json" → "/Projects/Lib/Catalogues/bsi-grundschutz.json"
Source file located in:           "/Projects/SSP/ssp-server.json"
→ Suggested relative path:        ../Lib/Catalogues/bsi-grundschutz.json
```

**If resolution is not possible:** Absolute paths whose files cannot be found even via the project index remain unchanged. mjEdit provides a note indicating which paths could not be resolved.

## The Correction Dialog

As soon as mjEdit finds at least one resolvable path, the interactive **"Unreachable Paths" dialog** appears instead of the simple warning message:

**Table with Preview:**

| Column | Content |
|---|---|
| **JSON Path** | Position of the href attribute in the OSCAL document |
| **Absolute Path** | Original, unreachable absolute path |
| **Suggested Relative Path** | Calculated relative path (empty if not resolvable) |
| **Status** | ✓ Resolvable (green) / ✗ Not resolvable (red) |

**Correction Modes (Radio Buttons):**

| Mode | Behaviour |
|---|---|
| **Keep paths (no changes)** | Close the dialog, open the file unchanged |
| **Correct only this file (Level 1)** | Set resolvable paths in the currently open file to relative paths |
| **Correct all files in the directory** | Check and correct all OSCAL JSON files in the source directory (Level 1 for each file) |
| **All files in the project directory** | Check all OSCAL JSON files in the project (Level 1+2 with pre-built project index) |

> **Tip:** The mode "All files in the project directory" is ideal after opening a migrated project for the first time. It converts all absolute path references in the entire project at once – afterwards, the project works on any computer without adjustments.

## Batch Processing with Progress Indicator

In directory or project mode, mjEdit starts a background process with a progress indicator:

- **Progress bar** displays `File 12/47: ssp-server.json`
- **Cancel button** stops processing after the next file
- **Completion message** shows: Checked files / modified files / total corrected paths

**Skipped directories:** System directories such as `.venv`, `.git`, `__pycache__`, `node_modules` are automatically excluded.

## Recommended Workflow for Project Migration

```
1. Copy the project directory to the new computer
       ↓
2. Open the first OSCAL document in mjEdit
       ↓
3. Path correction dialog appears (if absolute paths are detected)
       ↓
4. Select correction mode “All files in the project directory"
       ↓
5. Batch processing starts → Progress indicator
       ↓
6. Completion message: “42 files checked, 38 modified, 156 paths corrected"
       ↓
7. All OSCAL documents open and link correctly
```

## Why Relative Paths in OSCAL Are Better

| Property | Absolute Paths | Relative Paths |
|---|---|---|
| **Portability** | ✗ Machine-dependent | ☑ Usable everywhere |
| **Teamwork** | ✗ Every user needs the same path | ☑ Relative to the project folder |
| **Version Control** | ✗ Local paths in the repo | ☑ No environment dependencies |
| **Deployment** | ✗ Needs adjustment | ☑ Works on any system |
| **OSCAL Best Practice** | Not recommended | ☑ NIST recommendation for portable documents |

## Theme System

| Theme | Description |
|---|---|
| **Dark** | Professional dark mode (default), high contrast |
| **Light** | Bright interface for daylight workstations |
| **Blue** | Blue-accented design |
| **Green** | Green-accented design |

- File colours in the tree configurable per file extension
- Live switching without restart
- Extendable via QSS (Qt StyleSheets)

## Print and Export

| Format | Available from |
|---|---|
| **Print** | Text tab, Markdown tab, PDF tab (with page setup and preview) |
| **PDF Export** | Markdown tab, OSCAL tabs (with title, author, header/footer, font selection) |
| **HTML Export** | Markdown tab |
| **XML Export** | Text tab (JSON → XML) |
| **CSV Export** | Text tab (JSON → CSV), SSP tab (inventory) |
| **Markdown Export** | All OSCAL tabs |

## Internationalisation

| Language | Status |
|---|---|
| German | ☑ Complete (source language) |
| English | ☑ Fully translated |

- Language switching without restart
- All GUI texts, error messages, and menus translated
- Expandable for additional languages (gettext-based)

## Global Shortcuts

| Shortcut | Action |
|---|---|
| Ctrl+N | New file |
| Ctrl+O | Open file |
| Ctrl+S | Save |
| Ctrl+Shift+S | Save as |
| Ctrl+Q | Quit |
| Ctrl+Tab / Ctrl+Shift+Tab | Next / Previous tab |
| Ctrl+W | Close tab |
| F6 | Switch focus (Tree ↔ Editor) |
| Ctrl+P | Print |
| Ctrl+Shift+B | Create backup |
| Ctrl+< | Insert snippet |
| F1 | Open help |

All shortcuts can be freely configured in `keybindings.json`.

---

# 6. Plugin System in Detail

mjEdit is fundamentally **modular** in design. Instead of embedding all functions directly into the core, mjEdit outsources specialised capabilities to interchangeable **plugins** – following the principle of **functional modularity**. The result: A lightweight core application that only loads what you actually need.

## 6.1 Why a Plugin System?

| Advantage | Description |
|---|---|
| **Slimmer Start** | Only activated plugins are loaded – less RAM, faster start-up |
| **Functional Modularity** | Each plugin encapsulates exactly one domain (OSCAL, browser, MCP, database) – clearly separated responsibilities |
| **Extensibility** | Add new features as plugins without changing the core code |
| **Custom Configuration** | Each user activates only the plugins they need |
| **Independent Updates** | Plugins can be updated individually without re-releasing the entire application |
| **Developer-Friendly** | Create your own plugins with the `BasePlugin` class, hook system, and service facade in just a few lines |

## 6.2 Technical Architecture

The plugin system is based on three pillars:

**1. Three-phase Lifecycle – Security through Controlled Loading**

Each plugin goes through three clearly defined phases:

| Phase | Method | Timing | What Happens |
|---|---|---|---|
| **1. Loading** | `on_load()` | Before GUI initialisation | Register hooks, define menus. GUI is **not** available yet! |
| **2. GUI Ready** | `on_gui_ready()` | After GUI setup | Create tabs, initialise widgets, access MainGUI |

| Phase | Method | Timing | What Happens |
|---|---|---|---|
| **3. Unloading** | `on_unload()` | On exit | Release resources, remove menus, deregister hooks |

This separation prevents race conditions and ensures that plugins never access an incomplete GUI.

**2. Hook-based Communication – Loose Coupling Instead of Tight Dependencies**

Plugins communicate with the core application and with each other via an **event hook system**. Instead of using direct method calls, plugins register callbacks for specific events:

```
# Plugin registers for the "File Opened" event
self.register_hook("file_opened", self.on_file_opened)
```

When mjEdit opens a file, all registered callbacks are notified – with automatic error isolation: if a plugin crashes, all others continue running.

**Available hooks:** `file_opened`, `file_saved`, `file_renamed`, `save_active_plugin_tab`, `add_menu`, `add_toolbar`, `open_external_url`, `on_tab_changed`, and more.

**3. Service Facade – Controlled Access to the Core Application**

Plugins do not access GUI internals directly but via a `PluginCoreServices` facade. This reduces coupling and makes plugins more robust against internal changes:

```
main_gui = self.get_main_gui()          # MainGUI instance
self.services.set_status("Loaded")      # Set status bar
self.services.log("Plugin active", "info") # Logging with plugin name
```

## 6.3 Plugin Manager – Managing Plugins

The integrated **Plugin Manager** (accessible via the *Tools → Plugin Manager* menu) provides a convenient interface for managing all plugins:

| Function | Description |
|---|---|
| **Enable** | Tick the checkbox – the plugin will be loaded on the next start |
| **Disable** | Untick the checkbox – the plugin will no longer be loaded, saving resources |
| **Plugin Details** | View name, version, type (GUI/Editor/Tool), and description |

| Function | Description |
|---|---|
| **Update** | Rescan the plugin directory (e.g., after manually adding a plugin) |
| **Restart** | Saves all open files and restarts mjEdit with the new plugin configuration |

**How to install a new plugin:**

1. Copy the plugin folder (containing `__init__.py` and `plugin.py`) into the `plugins/` directory
2. Open the Plugin Manager (*Tools → Plugin Manager*)
3. Click **Update** – the new plugin will appear in the list
4. Enable the plugin by ticking the checkbox and clicking **Save**
5. Restart mjEdit (using the **Restart** button or manually)

**How to disable an unnecessary plugin:**

1. Open the Plugin Manager
2. Untick the checkbox for the desired plugin
3. Click **Save** and restart mjEdit

Unnecessary plugins (e.g., the database plugin or the example plugin) can be disabled at any time to save resources and keep the interface tidy.

> **Note for compiled versions (.exe):** In the standalone compiled version, all plugins are embedded in the application package. You can **enable and disable** plugins but cannot remove or add new ones afterwards. If you require a customised version with a specific set of plugins – for example, without the browser plugin or with industry-specific extensions – you can license a **tailored Professional version**. Contact us for details on custom builds.

## 6.4 Bundled Plugins

mjEdit is delivered with the following plugins:

| Plugin | Type | Default | Description |
|---|---|---|---|
| **OSCAL Compliance** | Editor | ☑ active | All 8 OSCAL document types with specialised tabs (including **OSCAL Mapping Tab** with offline AI matching) |

| Plugin | Type | Default | Description |
|---|---|---|---|
| **MCP Server** | Tool | ☑ active | 88 MCP tools for AI integration (STDIO + SSE) |
| **Browser** | Tool | ☑ active | Chromium-based web browser with bookmarks |
| **Transform-Script** | Editor | ☑ active | QC script creation and conversion (F12) |
| **Network Discovery** | Tool | disabled | Scapy-based LAN scan (ARP/ICMP); imports hosts as OSCAL `inventory-items` directly into the active SSP tab |
| **Database** | Editor | disabled | Database connection for JSON data |
| **Example Plugin** | Editor | disabled | Template and documentation for custom plugin development |

---

## 6.5 OSCAL Plugin (Core Plugin)

The largest and most comprehensive plugin – specialised for all 8 OSCAL document types.

**Catalogue Functions:**

- Group navigation with tree structure
- Control display with full details (title, description, properties, parts)
- Control search and filter by ID, title, or content
- Add, edit, delete controls
- Nested group support (up to 3 levels)
- Import controls from other catalogues
- UUID auto-generation (RFC 4122)
- Metadata management (title, version, date)
- Back-matter resources (Base64 attachments)
- Cascade selection for entire groups via MCP

**Profile Functions:**

- Profile creation with metadata
- Baseline selection from imported catalogues
- Modifications: additions, deletions, replacements
- Parameter tuning for controls
- Profile resolution (resolved catalogue)
- Profile merge (combine multiple profiles)
- Exclusion lists
- Save-as dialogue with target directory selection

**SSP Functions:**

- System definition with metadata
- Component management (software, hardware, service, etc.)
- Inventory management (specific assets with properties)
- Document control implementation (by component)
- Responsibility assignment (roles and parties)
- Network discovery (automatic IP, hostname, MAC)
- CSV import/export for inventory
- Network diagram generation (NWDiag)
- Compliance check (analyse control coverage)
- Impact analysis (CVE → affected components)
- Dynamic documentation (auto-update with inventory changes)

**Assessment Plan Functions:**

- Define assessment activities
- Select assessment subjects
- Set scope and timeframe
- Assign test methods
- Schema-compliant mandatory fields (activities, subjects, back-matter)

**Assessment Results Functions:**

- Record findings with severity and evidence
- Document observations and notes
- Link evidence to resources
- Report generation

**POA&M Functions:**

- Create remediation items with target dates
- Risk prioritisation
- Milestone tracking

- Status management (planned, in progress, completed)
- Resource allocation

**Component Definition Functions:**

- Create reusable component libraries
- Component types: software, hardware, service, policy, etc.
- Document control implementation per component

**Mapping Collection Functions:**

- Bidirectional control mapping between frameworks
- **Offline AI matching** with a local multilingual sentence-transformer model (no cloud, no data sharing)
- Three methods: `syntactic`, `semantic`, `auto` (score fusion)
- NIST ↔ internal format adapter
- Visual highlighting of linked controls
- Auto-suggest with configurable confidence threshold
- Auto-accept mode for bulk adoption of high-confidence suggestions
- Generate derived OSCAL documents (resolved profile, SSP proposal, component definition stub, POA&M stub) from the mapping
- Markdown export of the mapping table (cross-reference, gap analysis)
- Status support (complete, not-complete, draft, deprecated, superseded as per OSCAL 1.2.1)

## 6.6 MCP-Server-Plugin

Provides mjEdit as an MCP server for AI systems.

**Server functions:**

- STDIO and SSE transport
- Auto-start on application launch (configurable)
- Rate limiting (100 requests/minute, configurable)
- Path whitelist for file access
- Audit logging of all tool calls
- Activity monitor for real-time communication display
- Tool pagination (20 per page) for LLM token limits
- Pre-validation sanitizer (parameter correction before schema validation)
- Non-blocking communication
- SSE reconnection with exponential backoff

## 6.7 Browser Plugin

Integrated web browser as a tab in mjEdit.

- URL navigation with address bar
- Bookmark management
- HTTPS status display
- Multiple browser tabs
- Lazy loading (WebEngine only when needed)
- Configurable homepage
- Automatically open external URLs from Markdown

## 6.8 Transform-Script-Plugin

QC script transformation for qFORM forms.

- GUI dialogue for QC script creation
- Python → QC string conversion
- Syntax highlighting in the dialogue
- QC validation
- Helper functions: `verify()`, `parse()`, `is_qc_string()`

## 6.9 Database Plugin

Database integration for JSON data.

- Configure database connections
- Load JSON from the database
- Save JSON to the database
- Bidirectional synchronisation
- Database viewer
- SQL query tool

## 6.10 Network Discovery Plugin — LAN Inventory Directly in the SSP

The **Network Discovery Plugin** addresses one of the most time-consuming gaps in any SSP project: capturing the actual IT inventory. The plugin scans the local network via ARP sweep (scapy-based) and transfers the discovered hosts — including IP address, MAC address, FQDN, and hostname — as OSCAL v1.2.1-compliant `inventory-item` entries **directly** into an open SSP tab.

**Architecture and Security:**

- **scapy** as a network library (raw sockets); no `nmap`, no shell calls, no external service
- Subnet size limited to **a maximum of 4096 IP addresses** (anti-DoS)
- **Confirmation dialog** before each scan with the note "Scan only in your own networks"
- **Reverse DNS** with a 1-second timeout per host, parallelised
- On Windows, Npcap is additionally required as a driver (raw socket access)
- The plugin is **disabled by default** and must be explicitly activated in the plugin manager — no background scans without user action

**How it integrates with SSP documents:**

An SSP (System Security Plan) contains in the `system-implementation.inventory-items` section the list of all assets assigned to the system (servers, switches, endpoints, IoT components, etc.). In traditional compliance projects, this list is manually compiled from Excel inventory lists or CMDBs — error-prone, quickly outdated, often incomplete. The Network Discovery Plugin enables a **live reconciliation**:

1. Work in the OSCAL SSP tab of the target system.
2. Open the scan dialog via **Plugins** → 🔎 **Network Discovery…**.
3. Enter the subnet (CIDR, e.g., `10.0.5.0/24`) and timeout, then **start the scan**.
4. In the results table, select/deselect the hosts to be included in the SSP.
5. Select the target SSP from the dropdown (all open SSP tabs are listed); optionally activate "Skip duplicates (same IP)" to avoid overwriting existing data.
6. Click **"Add to SSP"** — the selected hosts are appended to `system-implementation.inventory-items`; the SSP tab is marked as modified (`*` in the tab title) and can be saved with `Ctrl+S`.

**OSCAL Mapping of Scan Results** (each host becomes an `inventory-item`):

| Scan Result | OSCAL Field |
|---|---|
| IP address | `props[].name="ipv4-address"` |
| MAC address | `props[].name="mac-address"` |
| FQDN (fully qualified domain name) | `props[].name="fqdn"` |
| Hostname (short form) | `props[].name="hostname"` |
| Scan timestamp | `props[].name="discovered-at"` |

Each entry also receives an RFC-4122-compliant UUID, an empty `description` field for manually adding the asset function, and a default empty `responsible-parties` list for later assignment of responsibilities.

**Use Case Example:**

*Scenario:* A consulting firm is creating an SSP for a client based on BSI IT-Grundschutz for a newly commissioned subnet `192.168.50.0/24`. Manual inventory would take several hours (switch configurations, ARP tables, manual entry).

*With the plugin:* The consultant opens the prepared SSP tab, initiates a scan on `192.168.50.0/24` (duration

approx. 30 seconds), receives a list of 47 hosts including hostnames from the internal DNS, selects 41 productive systems (excluding 6 test VMs), and adds them to the SSP with one click. Subsequently, the `description` and `responsible-parties` fields for each host are completed in the SSP tab — the asset inventory is built in **under 15 minutes**, fully OSCAL-compliant and audit-ready.

*Repeat for the next audit:* The same scan in the same subnet, with the "Skip duplicates (same IP)" option enabled, identifies **new hosts** (asset drift) and highlights them — providing the basis for systematic inventory maintenance throughout the system lifecycle.

> **Screenshot Note 1:** *Discovery dialog: at the top, a CIDR input field with validation notice ("Maximum 4096 IPs"), timeout spinbox, next to it a "Start Scan" button in the primary colour; in the middle, the results table with columns Checkbox, IP, MAC, Hostname, FQDN; at the bottom, the target SSP dropdown and "Add to SSP" button.*

> **Screenshot Note 2:** *Confirmation dialog before the scan with a clear warning symbol and text "Start ARP scan on 192.168.50.0/24 (256 IPs)? Only scan your own networks!" — two buttons "Cancel" and "Proceed" (the latter highlighted in red).*

> **Screenshot Note 3:** *SSP tab after adding: in the `inventory-items` list, 41 newly added entries (highlighted in green), each expanded with properties `ipv4-address`, `mac-address`, `fqdn`, `hostname`, and the UUID; tab title with `*` indicating unsaved changes.*

**Advantages over External Inventory Tools:**

- **No data exfiltration:** Scan results remain entirely within mjEdit — no transfer to external CMDBs, no cloud connection.
- **Direct SSP integration:** Unlike general network scanners, the plugin writes not to CSV or a database but to the **correct OSCAL field** of the **correct SSP document**.
- **Reproducibility:** Scan configuration (CIDR, timeout) is recorded along with the timestamp in the `inventory-item` `prop` — auditors can repeat the scan at any time.
- **Schema validation:** Before saving the SSP, the document, including new inventory items, is validated against the OSCAL 1.2.1 schema.

## 6.11 Plugin Development

mjEdit offers an open plugin system for custom extensions:

- Base class `BasePlugin` with full lifecycle
- Hook system for event-based integration
- Menu and toolbar integration
- Tab registration
- Configuration access

- Example plugin as a template

# 7. Template and Snippet System

mjEdit features a two-tiered template system that is consistently integrated throughout the entire application: **Templates** for complete files and **Snippets** for reusable text blocks. Both systems are file-based, ready to use immediately, and can be extended without programming – simply place a new file in the appropriate folder.

## 7.1 Templates – Creating New Files from Templates

A template is a complete template file that serves as a starting point for a new document. mjEdit copies the template, renames the copy, and opens it in the appropriate tab – ready for editing.

**Usage:**

| Method | Description |
|---|---|
| **Ctrl+N → "From Template"** | Select the fourth option in the "New File" dialog |
| **File Tree → Context Menu** | Right-click on a folder → "New File from Template" |
| **OSCAL Tab → Context Menu** | Element-specific templates directly in the OSCAL editor |
| **MCP-KI Client** | `template_list`, `template_create`, `editor_insert_template` |

**The Template Selection Dialog** (Ctrl+N → "From Template") is structured as a three-step workflow:

1. **Select a Template** – On the left, a tree with all templates from `data/templates/`, on the right, a live preview of the file contents. Folders are displayed with icons, files by type ( JSON, Markdown, Text).
2. **Set the Target Directory** – A second tree shows the project directories. The context menu allows you to create, rename, or delete subfolders.
3. **Assign a File Name** – The template name is suggested, automatically appended with the suffix `_new`. A colour indicator signals whether the name is already taken (red) or available (green).

**Intelligent File Recognition:** After creation, mjEdit automatically opens the new file in the correct tab:

- `.md` files → Markdown Tab
- qFORM files (JSON array with `"type": "qFORM"`) → Form Tab
- OSCAL files (with `catalog`, `profile`, `ssp` etc.) → OSCAL Tab
- All others → Text Tab

**Save as Template:** Any editor content can be added directly to the template library via Right-click → "Save as Template." The dialog offers directory navigation, file name and type selection (.json/.md/.txt), as well as JSON validation when saving.

## 7.2 OSCAL Templates – Context-Based Filtering by Keywords

The core of the template system for OSCAL users is **strict filename filtering**. When you add a new element via a template in an OSCAL tab, mjEdit shows **only** the templates whose filenames match the current context – not the entire library.

**The principle:** The array type (e.g. `roles`, `parties`, `inventory-items`) is matched against the filenames in the template directory. Only files containing the search term in their name appear in the dialog.

**Example:**

| Context in the OSCAL Tab | Displayed Templates |
|---|---|
| Add a new **role** | `roles-template-users.json`, `roles-template23.json` |
| Add a new **inventory item** | `inventory-item-basic-template.json`, `inventory-item-server-template.json`, `inventory-item-network-template.json`, `inventory-item-software-template.json` |
| Add a new **party** (person/organisation) | `parties-person-template.json`, `parties-organization-template.json` |
| Add a new **information type** | `information-types-fips199-high.json`, `information-types-grundschutz-personenbezogen.json`, `information-types-nist-pii.json` |
| Add a new **security impact level** | `security-impact-level-grundschutz.json`, `security-impact-level-iso27001.json`, `security-impact-level-nist-800-53.json` |

**Why this filtering is important:** OSCAL includes over 25 different array types (roles, parties, locations, controls, implementations, protocols, properties, etc.). Without filtering, the user would have to find the correct one in a list of 73+ templates. The filename convention (`{type}-template.json`) automates and extends the assignment: custom templates simply follow the same naming convention.

The following table shows all array types supported by the template system along with their meaning in the OSCAL context:

## Metadata Arrays (in all OSCAL document types)

| Array Type | OSCAL Context | Description |
|---|---|---|
| `roles` | Metadata | Roles such as “System Owner”, “Authorizing Official” or “ISB” – define responsibilities in the security process |
| `parties` | Metadata | Organisations and individuals involved in the system (operators, service providers, auditors) |
| `locations` | Metadata | Physical locations (data centres, offices, cloud regions) with address and contact details |
| `responsible-parties` | Metadata | Assignment of parties to roles – who fulfils which role in a specific case |
| `document-ids` | Metadata | Document identifiers (DOI, internal IDs) for unique referencing |
| `revisions` | Metadata | Document revision history with date, author and description |

## Component Definition and Implementation

| Array Type | OSCAL Context | Description |
|---|---|---|
| `props` | All levels | Key-value properties for arbitrary metadata on any OSCAL object |
| `links` | All levels | References to external resources, documents, or other OSCAL objects via URI |
| `responsible-roles` | Components, SSP | Which roles are responsible for a component or implementation |
| `protocols` | Components | Network protocols (HTTPS, SSH, LDAP) with port ranges used by a component |
| `control-implementations` | Components | Groups of implemented controls that a component fulfils – the core of the compliance statement |

| Array Type | OSCAL Context | Description |
|---|---|---|
| `implemented-requirements` | SSP, Components | Individual implemented requirements with a description of how a control is implemented |
| `set-parameters` | Profile, SSP | Set parameter values for controls (e.g., "Password length = 12 characters") |
| `statements` | SSP | Detailed implementation descriptions per control statement (a, b, c, ...) |

### System Security Plan (SSP)

| Array Type | OSCAL Context | Description |
|---|---|---|
| `inventory-items` | SSP | IT inventory: servers, workstations, network devices, software – each individual system with attributes |
| `information-types` | SSP | Classification of processed data based on protection needs (FIPS 199, BSI IT-Grundschutz, NIST) |
| `security-impact-level` | SSP | Security classification of the overall system (confidentiality, integrity, availability) |
| `system-status` | SSP | Operational status: operational, under-development, under-major-modification, disposition |
| `users` | SSP | User groups of the system with access rights and permission levels |
| `components` | SSP | Components used in the system (hardware, software, services, policies) |
| `by-components` | SSP | Mapping: which component implements which control – the compliance matrix |

| Array Type | OSCAL Context | Description |
|---|---|---|
| `leveraged-authorizations` | SSP | References to already authorised systems (e.g., cloud providers with their own ATO) |

### Assessment and Plan of Action

| Array Type | OSCAL Context | Description |
|---|---|---|
| `activities` | Assessment Plan | Planned assessment activities: interviews, tests, document reviews |
| `findings` | Assessment Results, POAM | Identified deviations and weaknesses from the audit |
| `observations` | Assessment Results | Individual observations (evidence) from the assessment |
| `risks` | Assessment Results, POAM | Identified risks with evaluation and classification |
| `poam-items` | POAM | Actions with deadlines and responsible parties to address findings |

> **Tip:** Creating your own templates for new array types is simple: Place a JSON file with the type keyword in the filename (e.g. `inventory-item-firewall-template.json`) in the folder `data/templates/OSCAL/elements/` – the dialog will find it automatically.

**Additional Features of the OSCAL Template Dialog:**

- **Recursive Search** in all subdirectories of `data/templates/OSCAL/`
- **Template Mapping** with 25+ predefined mappings for standard types
- **Preview** with automatic cleanup of `_comment` fields
- **Path Persistence** – the last selected folder is saved in the configuration
- **Edit After Insert** – optional checkbox opens the inserted element directly in the edit dialog
- **UUID Generation** – placeholder UUIDs in the template are automatically replaced with valid UUIDv4 values

**Available OSCAL Tabs with Template Support:**

| OSCAL Tab | Template Functions |
|---|---|
| **Catalogue** | New control, new group, subcontrol from template |
| **Component Definition** | New component, new property from template |
| **System Security Plan** | Inventory items, information types, security impact level, system status |
| **Assessment Plan** | Array elements (roles, parties, locations, activities) |
| **Assessment Results** | Array elements (findings, observations, risks) |
| **Plan of Action** | Array elements (findings, risks, poam-items) |
| **Profile** | Imports, alterations, parameter settings |

## 7.3 Supplied Template Library

mjEdit comes with over **73 OSCAL templates**, all verified for compliance with OSCAL v1.2.1:

| Category | Quantity | Examples |
|---|---|---|
| **Document Types** | 6 | Catalogue, Profile, SSP, Assessment Plan/Results, POAM |
| **Components** | 13 | Software, Hardware, Service, Network, Policy, Guidance, Infrastructure, Plan, Process, Standard, System, Validation, Interconnection |
| **Elements** | 43 | Controls, Groups, Roles, Parties, Inventory Items, Information Types, Security Impact Levels, Properties, Links, Protocols, Statements |
| **Metadata** | 5 | Document IDs, Locations, Parties, Responsible Parties, Roles |
| **Mapping** | 1 | Mapping collection for framework mappings |

Additionally, there are templates for **JSON** (6), **Markdown** (1), **qFORM** (2), and **QC Scripts** (1).

## 7.4 Snippets – Insert Text Blocks at the Cursor Position

A snippet is a reusable text block that is inserted at the current cursor position – ideal for frequently needed JSON structures, Markdown fragments, or OSCAL elements.

**Invocation:**

| Method | Description |
|---|---|
| **Ctrl+<** | Open the snippet dialog (configurable shortcut) |
| **Right-click → "Insert Snippet"** | In the text or Markdown tab |
| **MCP-KI Client** | `markdown_insert_snippet` |

**The Snippet Dialog** displays the directory tree from `data/snippets/` on the left with automatically expanded folders and a live preview of the selected snippet on the right. Navigate using the arrow keys, insert with Enter or double-click. The dialog remembers the last used file (`sys_snippet_last_file`).

**Included Snippets:**

| Category | Snippets | Description |
|---|---|---|
| **JSON** | `adresse_leer.json`, `person_leer.json` | Empty structure templates |
| **Markdown** | `markdown_table.md`, `markdown_template.md` | Tables, page templates |
| **qFORM** | 5 snippets | New fields, annotations, medications, patient data |

**Save as Snippet:** By right-clicking → "Save as Snippet," any selected or complete editor content can be directly saved as a snippet. The dialog supports directory navigation, creating new folders, and JSON validation.

## 7.5 Template vs. Snippet – When Should I Use Which?

| Criterion | Template | Snippet |
|---|---|---|
| **Purpose** | Create a new file | Insert a text block |
| **Granularity** | Complete file | Fragment / excerpt |
| **Result** | New file in the file system | Text at the cursor position |
| **Typical Use Case** | New OSCAL document, new configuration | JSON object, Markdown table, OSCAL element |
| **Storage Location** | `data/templates/` | `data/snippets/` |
| **Shortcut** | Ctrl+N → "From Template" | Ctrl+< |

## 7.6 Creating Your Own Templates and Snippets

The system is entirely **file-based** – no database, no registry, no build process. Create new templates in three steps:

1. **Create a file** – Place a `.json`, `.md`, or `.txt` file in the appropriate subfolder
2. **Follow naming conventions** – For OSCAL templates: `{type}-{variant}-template.json` (e.g. `inventory-item-router-te`
3. **Done** – The template will immediately appear in the dialogue, and OSCAL templates will also automatically show up in the context-specific filter

**Alternatively:** Right-click in the editor → "Save as Template" or "Save as Snippet" – mjEdit handles folder

selection, naming, and saving via the dialogue.

**Directory structure:**

```
data/templates/                    data/snippets/
├── JSON/          # JSON templates ├── JSON/       # JSON snippets
├── MARKDOWN/      # MD templates   ├── MARKDOWN/   # MD snippets
├── OSCAL/         # OSCAL templates├── OSCAL/      # OSCAL fragments
│   ├── components/  # 13 types     ├── QFORM/      # qFORM fields
│   └── elements/    # 43 elements  └── QSCRIPT/    # QC script snippets
├── QFORM/         # Forms
└── QSCRIPT/       # QC scripts
```

## 7.7 AI Integration: Templates via MCP

Templates are also available to AI clients via the MCP server:

| MCP Tool | Function |
|---|---|
| `template_list` | List all available templates by category |
| `template_create` | Create a new template from AI-generated content |
| `template_delete` | Remove a template |
| `template_export` / `template_import` | Exchange templates between installations |
| `editor_insert_template` | Insert a template into the editor via fuzzy name search |
| `oscal_model_create` | Create an OSCAL document with template support |
| `oscal_model_list_templates` | List available OSCAL templates |

**MCP Resources** for snippets:

| Resource | Description |
|---|---|
| `mjedit://snippets/json` | JSON code snippets |
| `mjedit://snippets/markdown` | Markdown snippets |
| `mjedit://snippets/oscal` | OSCAL snippets |
| `mjedit://snippets/qform` | qFORM snippets |
| `mjedit://snippets/qscript` | QC script templates |
| `mjedit://templates/list` | All available templates |

# 8. Security and Quality

## 8.1 Application Security

| Measure | Detail |
|---|---|
| RestrictedPython Sandbox | User code runs without system access |
| Path Traversal Protection | `../` attacks are blocked |
| Null Byte Blocking | No null bytes in file paths |
| File Validation | Contents are checked before opening |
| No Forbidden Functions | `eval()`, `exec()`, `os.system()`, `pickle` are forbidden |
| MCP Rate Limiting | Sliding window against overload |
| MCP Path Whitelist | Only configured directories are accessible |
| Audit Logging | All MCP operations are logged |

## 8.2 Code Quality

| Metric | Value |
|---|---|
| Bandit Security Audit | 0 Medium/High Issues |
| Implemented User Stories | 900+ Features |
| Automated Tests | 1350+ Tests (all green) |
| Documentation | Complete (DE + EN) |
| i18n Coverage | 100% of all GUI texts |

## 8.3 OSCAL Data Quality

- Pydantic models with runtime validation for every creation and modification
- JSON schema validation against official NIST OSCAL v1.2.1 schemas
- UUID format check (RFC 4122) with auto-correction
- Reference integrity check (UUID/Href/Control-ID)
- Duplicate protection for inventory component links

# 9. Supported AI Clients

mjEdit provides its MCP server via two transport protocols: **STDIO** (direct process communication) and **SSE** (HTTP Server-Sent Events). Depending on the AI client, one or both transports are supported.

## 9.1 Full Support (88 Tools + 22 Resources + 15 Prompts)

| Client | Transport | Configuration |
|---|---|---|
| **Claude Desktop** | STDIO | `claude_desktop_config.json` |
| **Cursor IDE** | STDIO | MCP server in Settings |
| **VS Code (GitHub Copilot)** | STDIO | `.vscode/mcp.json` |

## 9.2 AnythingLLM – Open-Source Reference Client for MCP

**AnythingLLM** holds a special position among the supported AI clients: As an **open-source platform** (MIT licence) with full MCP support, AnythingLLM is the **recommended reference MCP HOST client** for mjEdit – particularly for organisations that value data sovereignty, self-hosting, and RAG integration.

| Feature | Detail |
|---|---|
| **Licence** | MIT (Open Source) – free to use, including commercially |
| **MCP Transport** | SSE (HTTP) – ideal for Docker, network setups, and multi-user environments |
| **Deployment** | Desktop app (Windows, macOS, Linux) or Docker container |
| **RAG Knowledge Base** | Embed your own compliance documents and combine them with MCP tools |
| **LLM Selection** | Fully customisable: OpenAI, Anthropic Claude, Ollama (local), LM Studio, Azure, and more |
| **Offline Capability** | Yes – fully usable without internet with local models (e.g., Ollama) |
| **Multi-User** | Team workspaces with role management |

**Why AnythingLLM as the Reference Client?**

- **Open Source + Self-Hosted:** No dependency on cloud services, full control over data and infrastructure

- **RAG + MCP Synergy:** As the only supported client, AnythingLLM combines a RAG knowledge base (your own compliance documents) with mjEdit's 88 MCP tools – the AI understands your organisation and can simultaneously create OSCAL documents
- **Flexible LLM Backend:** From GPT-4o to Claude to free local models via Ollama – you decide which model processes your compliance data
- **Docker Integration:** Simple deployment as a container in the internal network, SSE transport via `host.docker.internal:8765`

**Configuration:**

| Variant | MCP URL | Specifics |
|---|---|---|
| **AnythingLLM Desktop** | `http://localhost:8765/sse` | Direct connection on the same machine |
| **AnythingLLM Docker** | `http://host.docker.internal:8765/sse` | Bridge network to the host system |

```
{
  "mcpServers": {
    "mjEdit": {
      "url": "http://localhost:8765/sse",
      "transport": "sse"
    }
  }
}
```

> **Tip:** AnythingLLM supports single-call MCP. Use the mjEdit tool `execute_steps` to bundle up to 20 tool calls in a single request – this saves tokens and significantly speeds up complex workflows.

For a detailed description of RAG integration and practical use cases, see AnythingLLM.

## 9.3 Example Configuration (Claude Desktop)

```
{
  "mcpServers": {
    "mjedit": {
      "command": "python",
      "args": ["C:/Path/to/mjEdit/src/main.py", "--mcp-stdio"]
    }
  }
}
```

# 10. Technical Specifications

## 10.1 System Requirements

| Requirement | Minimum |
|---|---|
| **Python** | 3.10+ |
| **Operating System** | Windows 10/11, Linux (Ubuntu 22.04+) |
| **RAM** | 512 MB (4 GB recommended for large catalogues) |
| **Hard Drive** | 200 MB + data |
| **GUI Framework** | PySide6 (Qt 6) |

## 10.2 Performance Metrics

| Operation | Typical Duration |
|---|---|
| Application start | 2–3 seconds (lazy loading) |
| Open JSON (10 KB) | < 100 ms |
| Open JSON (1 MB) | 1–2 seconds |
| Format JSON | < 500 ms |
| Search (10,000 entries) | < 200 ms |
| Load OSCAL catalogue | 2–5 seconds |
| Profile resolution | 1–3 seconds |
| Tree filter | < 100 ms |

## 10.3 Performance Optimisations

| Threshold | Optimisation |
|---|---|
| 100 KB | Performance mode activated |
| 200 KB | Spell check disabled |
| 1 MB | Syntax highlighting disabled |
| 1,000 controls | Progress display for OSCAL catalogues |

## 10.4 Pre-installed Compliance Data

| Dataset | Scope |
|---|---|
| **BSI IT-Grundschutz** | 111 groups, 2,128 controls, 444 parts, 817 sub-parts |
| **NIST SP 800-53** | 6 families, 468 controls |
| **Grundschutz++ Profile** | 489 controls |

## 10.5 Deployment

| Platform | Format |
|---|---|
| **Windows** | EXE via PyInstaller (including data) |
| **Linux** | AppImage |

## 10.6 Licence

**AGPL-3.0** – Free use, modification, and distribution under the terms of the GNU Affero General Public License v3.0.

# 11. Roadmap – Planned Features and Enhancements

| Feature | Description | Priority |
|---|---|---|
| **Bug fixes and stability improvements** | Continuous error correction and optimisation of the application | Very High |
| **Migration and specialisation of OSCAL dialogues** | Improved dialogues for editing OSCAL elements (controls, groups, components) with enhanced features such as standard GUI forms, inline editing, and validation | Very High |
| **Expansion of the template and text snippet library** | Additional pre-made templates and snippets for common use cases | Very High |
| **Optimisation of OSCAL-MAPPING** | Optimisation for the selection and processing of OSCAL controls in the mapping process | Very High |
| **GIT integration** | Team collaboration within the project | High |
| **Expansion and optimisation of the MCP API** | Additional tools, resources, and prompts for use by AI hosts | High |
| **Active AI REST API feedback channel to AnythingLLM** | Enables active communication with AnythingLLM via REST API from MCP-SERVER to MCP-HOST | High |
| **JIRA integration** | Integration with JIRA for task and project management | Medium |
| **Confluence integration** | Documentation and knowledge management with Confluence | Medium |
| **ServiceNow integration** | Connection to ServiceNow for ITSM workflows | Medium |
| **Dic2Pydantic migration of OSCAL-TABs** | Migration of OSCAL-TABs to Pydantic models for better validation and type safety | Low |

| Feature | Description | Priority |
| --- | --- | --- |
| **Integration with commercial 42tec tools** | Connection to commercial 42tec tools for enhanced functionalities in operational security and project management | Low |
| **Migration from Python to RUST programming** | Improved deployment and performance | Very Low |