Common Security Advisory Framework Version 2.1

Committee Specification Draft 02

25 February 2026

This stage:

https://docs.oasis-open.org/csaf/csaf/v2.1/csd02/csaf-v2.1-csd02.md (Authoritative)
https://docs.oasis-open.org/csaf/csaf/v2.1/csd02/csaf-v2.1-csd02.html
https://docs.oasis-open.org/csaf/csaf/v2.1/csd02/csaf-v2.1-csd02.pdf

Previous stage:

https://docs.oasis-open.org/csaf/csaf/v2.1/csd01/csaf-v2.1-csd01.md (Authoritative)
https://docs.oasis-open.org/csaf/csaf/v2.1/csd01/csaf-v2.1-csd01.html
https://docs.oasis-open.org/csaf/csaf/v2.1/csd01/csaf-v2.1-csd01.pdf

Latest stage:

https://docs.oasis-open.org/csaf/csaf/v2.1/csaf-v2.1.md (Authoritative)
https://docs.oasis-open.org/csaf/csaf/v2.1/csaf-v2.1.html
https://docs.oasis-open.org/csaf/csaf/v2.1/csaf-v2.1.pdf

Technical Committee:

OASIS Common Security Advisory Framework (CSAF) TC

Chair:

Justin Murphy (justin.murphy@mail.cisa.dhs.gov), DHS Cybersecurity and Infrastructure Security Agency
Omar Santos (osantos@cisco.com), Cisco Systems
Stefan Hagen (stefan@hagen.link), Individual

Editors:

Stefan Hagen (stefan@hagen.link), Individual
Thomas Schmidt (thomas.schmidt@bsi.bund.de), Federal Office for Information Security (BSI) Germany

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- Aggregator JSON schema:
  https://docs.oasis-open.org/csaf/csaf/v2.1/csd01/schema/aggregator.json.
  Latest stage: https://docs.oasis-open.org/csaf/csaf/v2.1/schema/aggregator.json.
- CSAF JSON schema: https://docs.oasis-open.org/csaf/csaf/v2.1/csd01/schema/csaf.json.
  Latest stage: https://docs.oasis-open.org/csaf/csaf/v2.1/schema/csaf.json.
- Meta JSON schema: https://docs.oasis-open.org/csaf/csaf/v2.1/csd01/schema/meta.json.
  Latest stage: https://docs.oasis-open.org/csaf/csaf/v2.1/schema/meta.json.
- Provider JSON schema: https://docs.oasis-open.org/csaf/csaf/v2.1/csd01/schema/provider.json.
  Latest stage: https://docs.oasis-open.org/csaf/csaf/v2.1/schema/provider.json.

Related work:

This specification replaces or supersedes:

- *Common Security Advisory Framework Version 2.0*. Edited by Langley Rock, Stefan Hagen, and Thomas Schmidt. 18 November 2022. OASIS Standard. https://docs.oasis-open.org/csaf/csaf/v2.0/os/csaf-v2.0-os.html. Latest stage: https://docs.oasis-open.org/csaf/csaf/v2.0/csaf-v2.0.html.

Declared JSON namespaces:

- https://docs.oasis-open.org/csaf/csaf/v2.1/schema/aggregator.json
- https://docs.oasis-open.org/csaf/csaf/v2.1/schema/csaf.json
- https://docs.oasis-open.org/csaf/csaf/v2.1/schema/meta.json
- https://docs.oasis-open.org/csaf/csaf/v2.1/schema/provider.json

Abstract:

The Common Security Advisory Framework (CSAF) Version 2.1 is the definitive reference for the language which supports creation, update, and interoperable exchange of security advisories as structured information on products, vulnerabilities and the status of impact and remediation among interested parties.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=csaf#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/csaf/.

This specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/csaf/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Key words:

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

Citation format:

When referencing this specification the following citation format should be used:

[csaf-v2.1]

Common Security Advisory Framework Version 2.1. Edited by Stefan Hagen, and Thomas Schmidt. 25 February 2026. OASIS Committee Specification Draft 02. https://docs.oasis-open.org/csaf/csaf/v2.1/csd02/csaf-v2.1-csd02.html. Latest stage: https://docs.oasis-open.org/csaf/csaf/v2.1/csaf-v2.1.html.

Notices

or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name ”OASIS” is a trademark of https://www.oasis-open.org/, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark/ for above guidance.

Table of Contents

1 Introduction

1.1 IPR Policy

This specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/csaf/ipr.php).

1.2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

For purposes of this document, the following terms and definitions apply:

Advisory reporting item that describes a condition present in an artifact and that requires action by the consumers.

Advisory Document artifact in which an analysis tool reports a result.

Advisory Management System software system that consumes the documents produced by analysis tools, produces advisories that enable engineering and operating organizations to assess the quality of these software artifacts at a point in time, and performs functions such as filing security advisories and displaying information about individual advisories. **Note**: An Advisory Management System can interact with a document viewer to display information about individual advisories.

Advisory Matching process of determining whether two advisories are targeting the same products and conditions.

Artifact sequence of bytes addressable via a URI. *Examples*: A physical file in a file system such as a source file, an object file, a configuration file or a data file; a specific version of a file in a version control system; a database table accessed via an HTTP request; an arbitrary stream of bytes returned from an HTTP request, a product URL, a common product enumeration value.

CSAF 2.0 to CSAF 2.1 Converter A CSAF Producer which takes a CSAF 2.0 Document as input and converts it into a valid CSAF 2.1 Document.

CSAF Additional Test A test that is not yet defined in section 6.

CSAF Asset Matching System program that connects to or is an asset database and is able to manage CSAF Documents as required by CSAF Management System as well as matching them to assets of the asset database.

CSAF Basic Validator A program that reads a document and checks it against the JSON schema and performs mandatory tests.

CSAF Consumer program that reads and interprets a CSAF Document.

CSAF Content Management System program that is able to create, review and manage CSAF Documents and is able to preview their details as required by CSAF Viewer.

CSAF Converter CSAF Producer that transforms the output of an analysis tool from its native output format into the CSAF format.

CSAF Core All parts of CSAF except for the parts covering extensions.

CSAF Direct Producer analysis tool which acts as a CSAF Producer.

CSAF Document security advisory text document in the format defined by this document.

CSAF Downloader A program that retrieves CSAF Documents in an automated fashion.

CSAF Extended Validator A CSAF Basic Validator that additionally performs recommended tests.

CSAF Extension A specified JSON object conveying additional information different from the content that can be conveyed with the CSAF Core elements.

CSAF Extension Additional Test A test whose execution depends on the presence of the specifying CSAF Extension that provides additional checks in the context of the CSAF Extension or the CSAF Document the extension is embedded in.

CSAF Extension Bundle A of compilation of machine-readable artifacts related to a single CSAF Extension.

CSAF Extension Collection A set of multiple CSAF Extension Packages.

CSAF Extension Overlay Test A test whose execution depends on the presence of the specifying CSAF Extension that extends or replaces a test standardized in this specification or a CSAF Additional Test.

CSAF Extension Package A of compilation of all artifacts related to a single CSAF Extension.

CSAF Extension Schema A JSON schema specifying the content and properties of a CSAF Extension.

CSAF Extension Specification The specification of a single CSAF Extension and related material.

CSAF Extension Test A test that is either a CSAF Extension Overlay Test or a CSAF Extension Additional Test.

CSAF Full Validator A CSAF Extended Validator that additionally performs informative tests.

CSAF Library A library that implements CSAF data capabilities.

CSAF Library with Basic Validation A CSAF Library that also satisfies the conformance target “CSAF Basic Validator”.

CSAF Library with Extended Validation A CSAF Library that also satisfies the conformance target “CSAF Extended Validator”.

CSAF Library with Full Validation A CSAF Library that also satisfies the conformance target “CSAF Full Validator”.

CSAF Management System program that is able to manage CSAF Documents and is able to display their details as required by CSAF Viewer.

CSAF Modifier CSAF Post-Processor which takes a CSAF Document as input and modifies the structure or values of properties. The output is a valid CSAF Document.

CSAF Post-Processor CSAF Producer that transforms an existing CSAF Document into a new CSAF Document, for example, by removing or redacting elements according to sharing policies.

CSAF Producer program that emits output in the CSAF format.

CSAF RVISC ID Updater A CSAF Post-Processor that updates vulnerability IDs in a given CSAF based on the entries in RVISC.

CSAF SBOM Matching System A program that connects to or is an SBOM database and is able to manage CSAF Documents as required by CSAF Management System as well as matching them to SBOM components of the SBOM database.

CSAF Superseder A CSAF Post-Processor that transforms a given CSAF into a superseded one.

CSAF Translator CSAF Post-Processor which takes a CSAF Document as input and translates values of properties into another language. The output is a valid CSAF Document.

CSAF Viewer CSAF Consumer that reads a CSAF Document, displays a list of the results it contains, and allows an end user to view each result in the context of the artifact in which it occurs.

CSAF Withdrawer A CSAF Post-Processor that transforms a given CSAF into a withdrawn one.

CVRF CSAF Converter CSAF Producer which takes a CVRF document as input and converts it into a valid CSAF Document.

Document output file produced by an analysis tool, which enumerates the results produced by the tool.

Driver tool component containing an analysis tool's or converter's primary executable, which controls the tool's or converter's execution, and which in the case of an analysis tool typically defines a set of analysis rules.

Embedded Link syntactic construct which enables a message string to refer to a location mentioned in the document.

Empty Array array that contains no elements, and so has a length of zero.

Empty Object object that contains no properties.

Empty String string that contains no characters, and so has a length of zero.

(End) User person who uses the information in a document to investigate, triage, or resolve results.

Engineering System software analysis environment within which analysis tools execute. **Note**: An engineering system might include a build system, a source control system, a result management system, a bug tracking system, a test execution system, and so on.

Extension tool component other than the driver (for example, a plugin, a configuration file, or a taxonomy).

External Property File file containing the values of one or more externalized properties.

Externalizable Property property that can be contained in an external property file.

Externalized Property property stored outside of the CSAF Document to which it logically belongs.

False Positive result which an end user decides does not actually represent a problem.

Filter refine a list by selecting entries that match given criteria.

Fingerprint stable value that can be used by a result management system to uniquely identify a result over time, even if a relevant artifact is modified.

Formatted Message

`message string which contains formatting information such as Markdown formatting ch`

Fully Qualified Logical Name string that fully identifies the programmatic construct specified by a logical location, typically by means of a hierarchical identifier.

Hierarchical String string in the format <component>{/<component>}*.

Line contiguous sequence of characters, starting either at the beginning of an artifact or immediately after a newline sequence, and ending at and including the nearest subsequent newline sequence, if one is present, or else extending to the end of the artifact.

Line (Number) 1-based index of a line within a file. **Note**: Abbreviated to "line" when there is no danger of ambiguity with "line" in the sense of a sequence of characters.

Localizable subject to being translated from one natural language to another.

Message String human-readable string that conveys information relevant to an element in a CSAF Document.

Nested Artifact artifact that is contained within another artifact.

Newline Sequence sequence of one or more characters representing the end of a line of text. **Note**: Some systems represent a newline sequence with a single newline character; others represent it as a carriage return character followed by a newline character.

Notification reporting item that describes a condition encountered by a tool during its execution.

Opaque neither human-readable nor machine-parsable into constituent parts.

Parent (Artifact) artifact which contains one or more nested artifacts.

Plain Text Message message string which does not contain any formatting information.

Plugin tool component that defines additional rules.

Policy set of rule configurations that specify how results that violate the rules defined by a particular tool component are to be treated.

Problem result which indicates a condition that has the potential to detract from the quality of the program. *Examples*: A security vulnerability, a deviation from contractual or legal requirements.

Product is any deliverable (e.g. software, hardware, specification, or service) which can be referred to with a name. This applies regardless of the origin, the license model, or the mode of distribution of the deliverable.

Property attribute of an object consisting of a name and a value associated with the name.

Redactable Property property that potentially contains sensitive information that a CSAF Direct Producer or a CSAF Post-Processor might wish to redact.

Reporting Item unit of output produced by a tool, either a result or a notification.

Reporting Configuration the subset of reporting metadata that a tool can configure at runtime, before performing its scan. *Examples*: severity level, rank

Repository container for a related set of files in a version control system.

Search compile a list of entries that match given criteria.

Taxonomy classification of analysis results into a set of categories.

Tag string that conveys additional information about the CSAF Document element to which it applies.

Text Artifact artifact considered as a sequence of characters organized into lines and columns.

Text Region region representing a contiguous range of zero or more characters in a text artifact.

Tool Component component of an analysis tool or converter, either its driver or an extension, consisting of one or more files.

Top-Level Artifact

`artifact which is not contained within any other artifact.`

Translation rendering of a tool component's localizable strings into another language.

Triage decide whether a result indicates a problem that needs to be corrected.

User see end user.

VCS version control system.

Vendor the community, individual, or organization that created or maintains a product (including open source software and hardware providers).

VEX Vulnerability Exploitability eXchange - enables a supplier or other party to assert whether or not a particular product is affected by a specific vulnerability, especially helpful in efficiently consuming SBOM data.

Viewer

`see CSAF Viewer.`

Vulnerability functional behavior of a product or service that violates an implicit or explicit security policy (conforming to ISO/IEC 29147 [ISO29147]).

White Space code point used to improve text readability or token separation as defined in section 12.2 of [ECMA-262].

XML eXtensible Markup Language - the format used by the predecessors of this standard, namely CVRF 1.1 and CVRF 1.2.

1.3 Normative References

[ECMA-262] *ECMAScript® 2024 Language Specification*, ECMA-262, 15th edition, June 2024, https://262.ecma-international.org/15.0/

[ISO8601-1] *Date and time — Representations for information interchange — Part 1: Basic rules*, International Standard, ISO 8601-1:2019(E), February 25, 2019, https://www.iso.org/standard/70907.html.

[JSON-Schema-Core] *JSON Schema: A Media Type for Describing JSON Documents*, draft-bhutton-json-schema-00, December 2020, https://datatracker.ietf.org/doc/html/draft-bhutton-json-schema-00.

[JSON-Schema-Validation] *JSON Schema Validation: A Vocabulary for Structural Validation of JSON*, draft-bhutton-json-schema-validation-00, December 2020, https://datatracker.ietf.org/doc/html/draft-bhutton-json-schema-validation-00.

[JSON-Hyper-Schema] *JSON Hyper-Schema: A Vocabulary for Hypermedia Annotation of JSON*, draft-handrews-json-schema-hyperschema-02, September 2019, https://json-schema.org/draft/2019-09/json-schema-hypermedia.html.

[Relative-JSON-Pointers] *Relative JSON Pointers*, draft-bhutton-relative-json-pointer-00, December 2020, https://datatracker.ietf.org/doc/html/draft-bhutton-relative-json-pointer-00.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, https://www.rfc-editor.org/info/rfc2119.

[RFC2606] Eastlake, D. 3rd and Panitz, A., "Reserved Top Level DNS Names", BCP 32, RFC 2606, DOI 10.17487/RFC2606, June 1999, https://www.rfc-editor.org/info/rfc2606.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, https://www.rfc-editor.org/info/rfc3339.

[RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-Separated Values (CSV) Files", RFC 4180, DOI 10.17487/RFC4180, October 2005, https://www.rfc-editor.org/info/rfc4180.

[RFC7230] Roy T. Fielding and Julian Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, https://www.rfc-editor.org/info/rfc7230.

[RFC7464] Williams, N., "JavaScript Object Notation (JSON) Text Sequences", RFC 7464, DOI 10.17487/RFC7464, February 2015, https://www.rfc-editor.org/info/rfc7464.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, https://www.rfc-editor.org/info/rfc8174.

[RFC8259] T. Bray, Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 8259, DOI 10.17487/RFC8259, December 2017, https://www.rfc-editor.org/info/rfc8259.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, https://www.rfc-editor.org/info/rfc9562.

[SPDX301] *The System Package Data Exchange® (SPDX®) Specification Version 3.0.1*, Linux Foundation and its Contributors, 2024, https://spdx.github.io/spdx-spec/.

1.4 Informative References

[CPE23-A] *Common Platform Enumeration: Applicability Language Specification Version 2.3 (NISTIR 7698)*, D. Waltermire, P. Cichonski, K. Scarfone, Editors, NIST Interagency Report 7698, August 2011, https://dx.doi.org/10.6028/NIST.IR.7698.

[CPE23-D] *Common Platform Enumeration: Dictionary Specification Version 2.3*, P. Cichonski, D. Waltermire, K. Scarfone, Editors, NIST Interagency Report 7697, August 2011, https://dx.doi.org/10.6028/NIST.IR.7697.

[CPE23-M] *Common Platform Enumeration: Naming Matching Specification Version 2.3*, M. Parmelee, H. Booth, D. Waltermire, K. Scarfone, Editors, NIST Interagency Report 7696, August 2011, https://dx.doi.org/10.6028/NIST.IR.7696.

[CPE23-N] *Common Platform Enumeration: Naming Specification Version 2.3*, B. Cheikes, D. Waltermire, K. Scarfone, Editors, NIST Interagency Report 7695, August 2011, https://dx.doi.org/10.6028/NIST.IR.7695.

[CSAF-v2.0] *Common Security Advisory Framework Version 2.0*. Edited by Langley Rock, Stefan Hagen, and Thomas Schmidt. 18 November 2022. OASIS Standard. https://docs.oasis-open.org/csaf/csaf/v2.0/os/csaf-v2.0-os.html. Latest stage: https://docs.oasis-open.org/csaf/csaf/v2.0/csaf-v2.0.html.

[CVE] *Common Vulnerability and Exposures (CVE) – The Standard for Information Security Vulnerability Names*, MITRE, 1999, Revised Feb. 2016, https://cve.mitre.org/docs/cve-intro-handout.pdf.

[CVE-NF] *Common Vulnerability and Exposures (CVE) – The Standard for Information Security Vulnerability Names - CVE ID Syntax Change*, MITRE, January 01, 2014, https://cve.mitre.org/cve/identifiers/syntaxchange.html.

[CVRF-1-1] *The Common Vulnerability Reporting Framework (CVRF) Version 1.1*, M. Schiffman, Editor, May 2012, Internet Consortium for Advancement of Security on the Internet (ICASI), https://www.icasi.org/the-common-vulnerability-reporting-framework-cvrf-v1-1/.

[CVRF-v1.2] *CSAF Common Vulnerability Reporting Framework (CVRF) Version 1.2*. Edited by Stefan Hagen. 13 September 2017. OASIS Committee Specification 01. https://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/cs01/csaf-cvrf-v1.2-cs01.html. Latest version: https://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csaf-cvrf-v1.2.html.

[CVSS2] *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*, P. Mell, K. Scarfone, S. Romanosky, Editors, First.org, Inc., June 2007, https://www.first.org/cvss/v2/cvss-v2-guide.pdf.

[CVSS30] *Common Vulnerability Scoring System v3.0: Specification Document*, FIRST.Org, Inc., June 2019, https://www.first.org/cvss/v3.0/cvss-v30-specification_v1.9.pdf.

[CVSS31] *Common Vulnerability Scoring System v3.1: Specification Document*, FIRST.Org, Inc., June 2019, https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf.

[CVSS40] *Common Vulnerability Scoring System v4.0: Specification Document*, FIRST.Org, Inc., June 18, 2024, https://www.first.org/cvss/v4-0/cvss-v40-specification.pdf.

[CWE] *Common Weakness Enumeration (CWE) – A Community-Developed List of Software Weakness Types*, MITRE, 2006, http://cwe.mitre.org/about/.

[CWE-20] *CWE - CWE-20: Improper Input Validation*, MITRE, https://cwe.mitre.org/data/definitions/20.html#Vulnerability_Mapping_Notes_20

[CWE-1023] *CWE - CWE-1023: Incomplete Comparison with Missing Factors*, MITRE, https://cwe.mitre.org/data/definitions/1023.html#Vulnerability_Mapping_Notes_1023

[CWE-A] *CWE - Archive*, MITRE, https://cwe.mitre.org/data/archive.html.

[CYCLONEDX161] *CycloneDX Software Bill-of-Material Specification JSON schema version 1.6.1*, cyclonedx.org, November 7, 2024, https://github.com/CycloneDX/specification/blob/1.6.1/schema/bom-1.6.schema.json.

[ECMA-427] *Package-URL (PURL) specification*, EMCA-427, 1st Edition, December 2025, https://ecma-international.org/wp-content/uploads/ECMA-427_1st_edition_december_2025.pdf

[EPSS] *Exploit Prediction Scoring System (EPSS)*, FIRST.Org, Inc., https://www.first.org/epss/

[FETCH] *Fetch: Living Standard*, https://fetch.spec.whatwg.org.

[GFMCMARK] *GitHub's fork of cmark, a CommonMark parsing and rendering library and program in C*, https://github.com/github/cmark.

[GFMENG] *GitHub Engineering: A formal spec for GitHub Flavored Markdown*, https://githubengineering.com/a-formal-spec-for-github-markdown/.

[ISO19770-2] *Information technology — IT asset management — Part 2: Software identification tag*, International Standard, ISO 19770-2:2015, September 30, 2015, https://www.iso.org/standard/65666.html.

[ISO29147] *Information technology — Security techniques — Vulnerability disclosure*, International Standard, ISO/IEC 29147:2018, October 23, 2018, https://www.iso.org/standard/72311.html.

[OPENSSL] *GTLS/SSL and crypto library*, OpenSSL Software Foundation, https://www.openssl.org/.

[PURL] *Package URL (PURL)*, GitHub Project, https://github.com/package-url/purl-spec.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, https://www.rfc-editor.org/info/rfc3552.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, https://www.rfc-editor.org/info/rfc3986.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, https://www.rfc-editor.org/info/rfc4122.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, https://www.rfc-editor.org/info/rfc4880.

[RFC7231] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, https://www.rfc-editor.org/info/rfc7231.

[RFC8322] Field, J., Banghart, S., and D. Waltermire, "Resource-Oriented Lightweight Information Exchange (ROLIE)", RFC 8322, DOI 10.17487/RFC8322, February 2018, https://www.rfc-editor.org/info/rfc8322.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, https://www.rfc-editor.org/info/rfc8615.

[RFC9116] Foudil, E. and Y. Shafranovich, "A File Format to Aid in Security Vulnerability Disclosure", RFC 9116, DOI 10.17487/RFC9116, April 2022, https://www.rfc-editor.org/info/rfc9116.

[RFC9535] S. Gössner, Ed., G. Normington, Ed., and C. Bormann, Ed., "JSONPath: Query Expressions for JSON", RFC 9535, DOI 10.17487/RFC9535, February 2024, https://www.rfc-editor.org/info/rfc9535.

[RVISC] *Registry for Vulnerability ID Systems for CSAF (RVISC)*, Part of the OASIS CSAF TC Registry, https://registry.csaf.dev/id/.

[RVISC-M] *Mapping for RVISC*, RVISC Mapping Part of the OASIS CSAF TC Registry, https://registry.csaf.dev/id/mapping/.

[RVISC-R] *Registry for RVISC*, RVISC Registry Part of the OASIS CSAF TC Registry, https://registry.csaf.dev/id/registry/.

[SCAP12] *The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.2*, D. Waltermire, S. Quinn, K. Scarfone, A. Halbardier, Editors, NIST Spec. Publ. 800-126 rev. 2, September 2011, https://dx.doi.org/10.6028/NIST.SP.800-126r2.

[SECURITY-TXT] Foudil, E. and Shafranovich, Y., *Security.txt Project*, https://securitytxt.org/.

[SemVer] *Semantic Versioning 2.0.0*, T. Preston-Werner, June 2013, https://semver.org/.

[SSVC] *SSVC: Stakeholder-Specific Vulnerability Categorization*, CERT/CC, https://certcc.github.io/SSVC/reference/

[SSVC-RNS] *Namespaces - SSVC: Stakeholder-Specific Vulnerability Categorization*, CERT/CC, https://certcc.github.io/SSVC/reference/code/namespaces/#registered-namespace

[SSVC-DP] *SSVC/data/json/decision_points at main · CERTCC/SSVC*, CERT/CC, https://github.com/CERTCC/SSVC/tree/main/data/json/decision_points

[VERS] *vers: a mostly universal version range specifier*, Part of the package-URL GitHub Project, https://github.com/package-url/vers-spec.

[VEX] *Vulnerability-Exploitability eXchange (VEX) - An Overview*, VEX sub-group of the Framing Working Group in the NTIA SBOM initiative, 27 September 2021, https://ntia.gov/files/ntia/publications/vex_one-page_summary.pdf.

[VEX-Justification] *Vulnerability Exploitability eXchange (VEX) - Status Justifications*, VEX sub-group of the Framing Working Group in the CISA SBOM initiative, June 2022, https://www.cisa.gov/sites/default/files/publications/VEX_Status_Justification_Jun22.pdf.

[XML] *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, T. Bray, J. Paoli, M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, W3C Recommendation, November 26, 2008, https://www.w3.org/TR/2008/REC-xml-20081126/. Latest version available at https://www.w3.org/TR/xml.

[XML-Schema-1] *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*, S. Gao, M. Sperberg-McQueen, H. Thompson, N. Mendelsohn, D. Beech, M. Maloney, Editors, W3C Recommendation, April 5, 2012, https://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/. Latest version available at https://www.w3.org/TR/xmlschema11-1/.

[XML-Schema-2] *W3C XML Schema Definition Language (XSD) 1.1 Part 2*: Datatypes W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, D. Peterson, S. Gao, A. Malhotra, M. Sperberg-McQueen, H. Thompson, Paul V. Biron, Editors, W3C Recommendation, April 5, 2012, https://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/. Latest version available at https://www.w3.org/TR/xmlschema11-2/.

1.5 Typographical Conventions

Keywords defined by this specification use this `monospaced` font.

```
Normative source code uses this paragraph style.
```

The information models of the CSAF schemata are illustrated in the prose in a generic YAML format in the shape of outlines. In contrast to extracts of JSON snippets in prior versions of this specification such YAML snippets can be annotated and validated both as being well-formed and matching the schema part. These outlines are directly extractable per JSON Paths (see [RFC9535] from the corresponding CSAF schema as documented in the normative format of this specification. The types used are the general `Mapping` (JSON `object`), `Sequence` (JSON `array`), and `String` types. The latter may be further constrained to specific facets like for example enumerations noted as `String.Enum`. General instances (like a CSAF document) are noted as such in angle brackets to emphasize the topology. Items of sequences are noted as YAML sequence items and annotated per comments naming the kind of sequence item they represent.

Some sections of this specification are illustrated with non-normative examples introduced with “Example” or “Examples” like so:

Example 1:

```
Informative examples also use this paragraph style but preceded by the text "Exampl
```

All examples in this document are informative only.

All other text is normative unless otherwise labeled e.g. like the following informative comment:

> This is a pure informative comment that may be present, because the information conveyed is deemed useful advice or common pitfalls learned from implementer or operator experience and often given including the rationale.

This document adheres to the Modern Language Association (MLA) style guidelines for formatting titles and terms.

2 Design Considerations

The Common Security Advisory Framework (CSAF) is a language to exchange Security Advisories formulated in JSON.

The term Security Advisory as used in this document describes any notification of security issues in products of and by providers. Anyone providing a product is considered in this document as a vendor, i.e. developers or maintainers of information system products or services. This includes all authoritative product vendors, Product Security Incident Response Teams (PSIRTs), and product resellers and distributors, including authoritative vendor partners. A security issue is not necessarily constrained to a problem statement, the focus of the term is on the security aspect impacting (or not impacting) specific product-platform-version combinations. Information on presence or absence of workarounds is also considered part of the security issue. This document is the definitive reference for the language elements of CSAF version 2.1. The encompassing JSON schema file noted in the Additional Artifacts section of the title page SHALL be taken as normative in the case a gap or an inconsistency in this explanatory document becomes evident. The following presentation in this section is grouped by topical area, and is not simply derivative documentation from the schema document itself. The information contained aims to be more descriptive and complete. Where applicable, common conventions are stated and known common issues in usage are pointed out informatively to support implementers of document producers and consumers alike.

This minimal required information set does not provide any useful information on products, vulnerabilities, or security advisories. Thus, any real-world Security Advisory will carry additional information as specified in section 3 Schema elements.

Care has been taken, to design the containers for product and vulnerability information to support fine-grained mapping of security advisories onto product and vulnerability and minimize data duplication through referencing. The display of the elements representing Product Tree and Vulnerability information has been placed in the sections named accordingly.

2.1 Construction Principles

A Security Advisory represented by a CSAF document is created through the application of multiple schemas during a process of collaboration between multiple stakeholders. The distinct and in part complex schemas are orchestrated, put into context and connected with each other serving the goal of providing actionable, structured, and validated information targeting a high degree of automation.

The format chosen is [JSONSchema] which allows validation and delegation to sub schema providers. The latter aligns well with separation of concerns and shares the format family of information interchange utilized by the providers of product and vulnerability information which migrated from XML to JSON since the creation of CSAF CVRF version 1.2, the pre-predecessor of this specification.

The acronym CSAF, “Common Security Advisory Framework”, stands for the target of concerted mitigation and remediation accomplishment.

Technically, the use of JSON schema allows validation and proof of model conformance (through established schema based validation) of the declared information inside CSAF documents.

The CSAF schema structures its derived documents into three main classes of the information conveyed:

1. The frame, aggregation, and reference information of the document
2. Product information considered relevant by the creator
3. Vulnerability information and its relation to the products declared in 2.

Wherever possible repetition of data has been replaced by linkage through ID elements. Consistency on the content level thus is in the responsibility of the producer of such documents, to link e.g. vulnerability information to the matching product.

A dictionary like presentation of all defined schema elements is given in the section 3. Any expected relations to other elements (linkage) is described there. This linking relies on setting attribute values accordingly (mostly guided by industry best practice and conventions) and thus implies, that any deep validation on a semantic level (e.g. does the CWE match the described vulnerability) is to be ensured by the producer and consumer of CSAF documents. It is out of scope for this specification.

Proven and intended usage patterns from practice are given where possible.

Delegation to industry best practices technologies is used in referencing schemas for:

- Classification for Document Distribution
  - Traffic Light Protocol (TLP)
    - Default Definition: https://www.first.org/tlp/
- Exploit Prediction
  - Exploit Prediction Scoring System (EPSS) [EPSS]
- Package Data
  - Package-URL (PURL) [ECMA-427]
- Platform Data
  - Common Platform Enumeration (CPE) Version 2.3 [CPE23-N]
- Vulnerability Categorization
  - Stakeholder-Specific Vulnerability Categorization [SSVC]
    - JSON Schema Reference: https://certcc.github.io/SSVC/data/schema/v2/SelectionList_2_0_0.schema.json
- Vulnerability Classification
  - Common Weakness Enumeration (CWE) [CWE]
    - CWE List: http://cwe.mitre.org/data/index.html
- Vulnerability Scoring
  - Common Vulnerability Scoring System (CVSS) Version 4.0 [CVSS40]
    - JSON Schema Reference: https://www.first.org/cvss/cvss-v4.0.2.json
  - Common Vulnerability Scoring System (CVSS) Version 3.1 [CVSS31]
    - JSON Schema Reference: https://www.first.org/cvss/cvss-v3.1.json
  - Common Vulnerability Scoring System (CVSS) Version 3.0 [CVSS30]
    - JSON Schema Reference: https://www.first.org/cvss/cvss-v3.0.json
  - Common Vulnerability Scoring System (CVSS) Version 2.0 [CVSS2]
    - JSON Schema Reference: https://www.first.org/cvss/cvss-v2.0.json

Even though not all - especially the referenced - JSON schemas prohibit specifically additional properties and custom keywords, it is strongly recommended not to use them. Suggestions for new fields SHOULD be made through issues in the TC's GitHub. The JSON schemas defined in this standard do not allow the use of additional properties and custom keywords.

> The standardized fields allow for scalability across different issuing parties and dramatically reduce the human effort and need for dedicated parsers as well as other tools on the side of the consuming parties.

Sole exception are the dedicated extension points (`x_extensions`) which allow the usage of extension defined outside this specification. To still ensure interoperability and guide tools how to deal with this data a CSAF Extension Content Schema and CSAF Extension Metaschema has been defined and MUST be adhered to (cf. section 2.4). Document issuers are advised to always balance pros and cons regarding the use of extension.

Section 4 defined profiles that are used to ensure a common understanding of which fields are required in a given use case. Additional conventions are stated in section 5. The tests given in section 6 support CSAF producers and consumers to verify rules from the specification which can not be tested by the schema. Section 7 states how to distribute and where to find CSAF documents. Safety, Security and Data Protection are considered in section 8. Finally, a set of conformance targets describes tools in the ecosystem.

2.2 Format Validation

The JSON schema 2020-12 dialect per default uses the `format` keyword just as annotation. To be able to ensure that the format constraints are validated as intended, the following metaschema is defined.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/meta.json",
  "$dynamicAnchor": "meta",
  "$vocabulary": {
    "https://json-schema.org/draft/2020-12/vocab/core": true,
    "https://json-schema.org/draft/2020-12/vocab/applicator": true,
    "https://json-schema.org/draft/2020-12/vocab/unevaluated": true,
    "https://json-schema.org/draft/2020-12/vocab/validation": true,
    "https://json-schema.org/draft/2020-12/vocab/meta-data": true,
    "https://json-schema.org/draft/2020-12/vocab/format-assertion": true,
    "https://json-schema.org/draft/2020-12/vocab/content": true
  },
  "allOf": [
    { "$ref": "https://json-schema.org/draft/2020-12/meta/core" },
    { "$ref": "https://json-schema.org/draft/2020-12/meta/applicator" },
    { "$ref": "https://json-schema.org/draft/2020-12/meta/unevaluated" },
    { "$ref": "https://json-schema.org/draft/2020-12/meta/validation" },
    { "$ref": "https://json-schema.org/draft/2020-12/meta/meta-data" },
    { "$ref": "https://json-schema.org/draft/2020-12/meta/format-assertion" },
    { "$ref": "https://json-schema.org/draft/2020-12/meta/content" }
  ]
}
```

> All vocabularies that can be used are defined in this metaschema as JSON schema 2020-12 dialect has the rule of non-inheritability of vocabularies.

It is then consequently used in all JSON schemas defined in this standard and replaces the reference to the JSON schema 2020-12.

```
  {
    "$schema": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/meta.json",
    // ...
  }
```

The format validation is enforced by setting the corresponding vocabulary as required.

> If a library used to parse, modify or create CSAF content is unable to deal with this meta schema, it could reach the objective by interpreting the schema as JSON schema 2020-12 dialect and enforcing the format validation via its implementation.

2.3 Date and Time

This standard uses the `date-time` format as defined in JSON Schema Draft 2020-12 Section 7.3.1. In accordance with [RFC3339] and [ISO8601-1], the following rules apply:

- The letter `T` separating the date and time SHALL be upper case.

- The separator between date and time MUST be the letter `T`.
- The letter `Z` indicating the timezone UTC SHALL be upper case.
- Fractions of seconds are allowed as specified in the standards mention above with the full stop (`.`) as separator.
- Empty timezones MUST NOT be used.
- The ABNF of RFC 3339, section 5.6 applies.

In contrast to the aforementioned standards, leap seconds MUST NOT be used.

> While a full support of [RFC3339] would be preferred, significant challenges have been mentioned by implementers as most libraries are lacking the support for leap seconds. To ensure interoperability, the decision was made to prohibit leap seconds.

2.4 Extensions

This standard allows for extensions to the standardized schema.

The following rules apply:

- An extension MUST NOT occur in any other place than specified.
- An extension MUST satisfy the Conformance Target "CSAF Extension".
- The schema specifying the content and properties of the CSAF Extension MUST satisfy the Conformance Target "CSAF Extension Schema".
- For official and registered extensions a CSAF Extension Package MUST be provided.
- CSAF Extensions SHOULD NOT provide CSAF Extension Overlay Tests for tests in the preset `extensions`.

2.4.1 Classes

There three classes of extensions:

1. Official extensions: These CSAF Extensions are specified by the OASIS CSAF TC and ensure interoperability.

   > Those extensions are used to backport upcoming features or integrate new features at the OASIS CSAF TC's discretion.

2. Registered extensions: These CSAF Extensions are specified by a named party, publicly available, well-documented and registered with the OASIS CSAF TC. Collisions are avoided through being listed in the register.

   > Those extension can be used for conveying additional information to a broader community and making it available for tool integration. The OASIS CSAF TC may or may not consider to replace such extension with a registered one or take it as input for a future version of CSAF.

3. Experimental extensions: These CSAF Extension can be used for tests and experiments. They SHOULD NOT be used in production.

2.4.2 Lists

The OASIS CSAF TC maintains:

- a list of official CSAF Extensions and their CSAF Extension Packages,
- a list of registered CSAF Extensions and their CSAF Extension Packages,
- a list of deprecated extensions, and
- a list of deny-listed extensions.

Deprecated extensions can still be used but support for them is removed in near future. If avoidable, they SHOULD NOT be used. Deny-listed extensions MUST NOT be used. The lists of deprecated extensions and deny-listed extensions MAY contain extensions that do not fulfill the conformance target CSAF Extension.

The list MAY contain additional examples.

2.4.3 Metaschema

The CSAF Extension Metaschema MUST be used to validate a CSAF Extension Schema.

2.4.4 Content Schema

An extension MUST contain exactly the following elements: CSAF Extension Schema (`$schema`), Extension Category (`category`), Critical (`critical`) and Content (`content`).

```
yaml$schema: String category: String.Enum content: Mapping critical: Boolean
```

> The CSAF Extension Content Schema works an interface definition. It is used in the CSAF Schema and defines a common structure for all CSAF Extensions.

2.4.4.1 Content Schema Property - Schema

CSAF Extension Schema (`$schema`) of value type CSAF Extension Content `$schema` Type (`content_schema_t`) contains the URL of the CSAF Extension JSON schema which the JSON object promises to be valid for. This SHOULD also be the location where the JSON schema can be retrieved. The value SHOULD match the `$id` of the JSON schema that defines the extension. The URL SHOULD contain a human-readable name for the extension before the version string. The versioning MUST use [SemVer]. URLs using a domain mentioned in [RFC2606] MUST be used according to their defined purpose.

Examples 1:

```
https://www.example.com/some-path/to-a-csaf-extension/schema/our-awesome-metric_1.0.0
https://oil-and-gas.isac.example/.well-known/csaf/extensions/schema/product-safety_17
```

2.4.4.2 Content Schema Property - Category

Extension Category (`category`) of value type `string` and `enum` holds the category of the extension content. Valid `enum` values are:

```
critical
high_value
informational
```

The value `critical` indicates, that the content provided through this extension is crucial to understand of the CSAF document this extension is included in. CSAF consumers and CSAF validators MUST warn if they process a CSAF Document including such extension instance and do not have the extension in question already implement.

The value `high_value` indicates, that the content provided through this extension is highly relevant and significantly aids in understanding the overall content of the CSAF document. CSAF consumers and CSAF validators SHOULD warn if they process a CSAF Document including such an extension instance and do not have the extension in question already implemented.

The value `informational` indicates, that the content provided through this extension just provides additional information. CSAF consumers and CSAF validators MAY warn if they process a CSAF Document including such an extension instance and do not have the extension in question already implemented.

2.4.4.3 Content Schema Property - Content

Content (`content`) of value type `object` contains the additional information in its properties. A Content object has at least `1` property. The property names (JSON keys) can be chosen freely; they SHOULD characterize the information given in its value.

2.4.4.4 Content Schema Property - Critical

Critical (`critical`) of value type `boolean` determines whether using the extension would fail a mandatory test. The `default` value for this is `false`.

For any failing test, a CSAF Extension Test MUST be provided.

2.4.5 Metadata

The CSAF Extension Metadata is a representation of metadata for a CSAF Extension. It is used to provide information for automated tools on certain requirements and incompatibilities the CSAF Extension has.

Examples 1:

```
https://www.example.com/some-path/to-a-csaf-extension/metadata/our-awesome-metric_1.0
https://oil-and-gas.isac.example/.well-known/csaf/extensions/metadata/product-safety_
```


3 Schema Elements

The CSAF schema describes how to represent security advisory information as a JSON document.

The CSAF schema Version 2.1 builds on the JSON Schema draft 2020-12 rules extended by the format validation enforcement (see 2.2).

```
"$schema": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/meta.json"
```

The schema identifier is:

```
"$id": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/csaf.json"
```

The further documentation of the schema is organized via Definitions and Properties.

- Definitions provide types that extend the JSON schema model
- Properties use these types to support assembling security advisories

Types and properties together provide the vocabulary for the domain specific language supporting security advisories.

The two mandatory properties are `$schema` and `document`. The three additional properties, `product_tree`, `vulnerabilities`, and `x_extensions` are optional.

3.1 Definitions

The definitions (`$defs`) introduce the following domain specific types into the CSAF language: Acknowledgments (`acknowledgments_t`), Branches (`branches_t`), Extensions (`extensions_t`), Full Product Name (`full_product_nam` Language (`lang_t`), Notes (`notes_t`), Product Group ID (`product_group_id_t`), Product Groups (`product_groups_t`), Product ID (`product_id_t`), Products (`products_t`), References (`references_t`), Subpath (`subpath_t`) and Version (`version_t`).

```
acknowledgments_t: Sequence
branches_t: Sequence
extensions_t: Sequence
full_product_name_t: Mapping
lang_t: String.Pattern
notes_t: Sequence
product_group_id_t: String
product_groups_t: Sequence
product_id_t: String
products_t: Sequence
references_t: Sequence
subpath_t: Mapping
version_t: String.Pattern
```

3.1.1 Acknowledgments Type

List of Acknowledgments (`acknowledgments_t`) type instances of value type `array` with `1` or more elements contain a list of `Acknowledgment` elements.

```
acknowledgments_t: Sequence
# ...
```

The value type of Acknowledgment is `object` with at least `1` and at most four properties. Every such element acknowledges contributions by describing those that contributed. The properties are: `names`, `organization`, `summary`, and `urls`.


```
acknowledgments_t:
- # <acknowledgment-instance>:
  names: Sequence
  organization: String
  summary: String
  urls: Sequence
# ...
```

3.1.1.1 Acknowledgments Type - Names

List of acknowledged names (`names`) of value type `array` with `1` or more items holds the names of contributors being recognized. Every such item of value type `string` with `1` or more characters represents the name of the contributor and contains the name of a single contributor being recognized.

Examples 1:

```
Albert Einstein
Johann Sebastian Bach
```

3.1.1.2 Acknowledgments Type - Organization

The contributing organization (`organization`) of value type `string` with `1` or more characters and holds the name of the contributing organization being recognized.

Examples 1:

```
CISA
Google Project Zero
Talos
```

3.1.1.3 Acknowledgments Type - Summary

Summary of the acknowledgment (`summary`) of value type `string` with `1` or more characters SHOULD represent any contextual details the document producers wish to make known about the acknowledgment or acknowledged parties.

Example 1:

```
First analysis of Coordinated Multi-Stream Attack (CMSA)
```

3.1.1.4 Acknowledgments Type - URLs

List of URLs (`urls`) of acknowledgment is a container (value type `array`) for `1` or more `string` of type URL that specifies a list of URLs or location of the reference to be acknowledged. Any URL of acknowledgment contains the URL or location of the reference to be acknowledged. Value type is `string` with format URI (`uri`).

3.1.1.5 Acknowledgments Type - Example

Example 1:

```
"acknowledgments": [
  {
    "names": [
      "Johann Sebastian Bach",
      "Georg Philipp Telemann",
      "Georg Friedrich Händel"
    ],
    "organization": "Baroque composers",
    "summary": "wonderful music"
  },
  {
    "organization": "CISA",
    "summary": "coordination efforts",
    "urls": [
      "https://cisa.gov"
    ]
  },
  {
    "organization": "BSI",
    "summary": "assistance in coordination"
  },
  {
    "names": [
      "Antonio Vivaldi"
    ],
    "summary": "influencing other composers"
  }
],
```

The above SHOULD lead to the following outcome in a human-readable advisory:

> We thank the following parties for their efforts:
>
> - Johann Sebastian Bach, Georg Philipp Telemann, Georg Friedrich Händel from Baroque composers for wonderful music
> - CISA for coordination efforts (see: https://cisa.gov)
> - BSI for assistance in coordination
> - Antonio Vivaldi for influencing other composers

3.1.2 Branches Type

List of branches (`branches_t`) with value type `array` contains `1` or more branch elements as children of the current element.

```
# ...
branches_t: Sequence
# ...
```

Every Branch holds exactly `3` properties and is a part of the hierarchical structure of the product tree. The properties `name` and `category` are mandatory. In addition, the object contains either a `branches` or a `product` property.


```
# ...
branches_t:
- # <branch-instance>:
  branches: $defs.branches_t
  category: String.Enum
  name: String
  product: $defs.full_product_name_t
# ...
```

`branches_t` supports building a hierarchical structure of products that allows to indicate the relationship of products to each other and enables grouping for simpler referencing. As an example, the structure MAY use the following levels: `vendor` -> `product_family` -> `product_name` -> `product_version`. It is recommended to use the hierarchical structure of `vendor` -> `product_name` -> `product_version` whenever possible to support the identification and matching of products on the consumer side.

3.1.2.1 Branches Type - Branches

List of branches (`branches`) has the value type `branches_t`.

3.1.2.2 Branches Type - Category

Category of the branch (`category`) of value type `string` and `enum` describes the characteristics of the labeled branch. Valid `enum` values are:

```
architecture
host_name
language
patch_level
platform
product_family
product_name
product_version
product_version_range
service_pack
specification
vendor
```

The value `architecture` indicates the architecture for which the product is intended.

The value `host_name` indicates the host name of a system/service.

The value `language` indicates the language of the product.

The value `patch_level` indicates the patch level of the product.

The value `platform` indicates the (CPU) platform for which the product is intended.

The value `product_family` indicates the product family that the product falls into.

The value `product_name` indicates the name of the product.

The value `product_version` indicates exactly a single version of the product. The value of the adjacent `name` property can be numeric or some other descriptor. However, it MUST NOT contain version ranges of any kind.

It is recommended to enumerate versions wherever possible. Nevertheless, the TC understands that this is sometimes impossible. To reflect that in the specification and aid in automatic processing of CSAF documents the value `product_version_range` was introduced. See next section for details.

The value `product_version_range` indicates a range of versions for the product. The value of the adjacent `name` property SHALL NOT be used to convey a single version.

> In most cases, product version ranges hinder the use of product identification helpers as they need to identify a product completely. In such case, the product identification and also therefore matching, independent whether that is against SBOMs or assets, solely relies on the categorized strings. This contradicts the goal of using unique product identifiers which would resolve the naming and identification issue for products and are a next major step towards more automation and providing only relevant information to users. Therefore, the use of product version ranges should be avoided, whenever possible.

The value `service_pack` indicates the service pack of the product.

The value `specification` indicates the specification such as a standard, best common practice, etc.

The value `vendor` indicates the name of the vendor or manufacturer that makes the product.

3.1.2.3 Branches Type - Name

Name of the branch (`name`) of value type `string` with `1` or more characters contains the canonical descriptor or 'friendly name' of the branch.

Examples 1:

```
10
365
Microsoft
Office
PCS 7
SIMATIC
Siemens
Windows
```

A leading `v` or `V` in the value of `name` SHOULD only exist for the categories `product_version` or `product_version_range` if it is part of the product version as given by the vendor.

3.1.2.3.1 Branches Type - Name under Product Version

If adjacent property `category` has the value `product_version`, the value of `name` MUST NOT contain version ranges of any kind.

Examples 1 (for `name` when using `product_version`):

```
10
17.4
v3
```

> The `product_version` is the easiest way for users to determine whether their version is meant (provided that the given ancestors in the product tree matched): If both version strings are the same, it is a match - otherwise not. Therefore, it is always recommended to enumerate product versions instead of providing version ranges.

Examples 2 (for `name` when using `product_version` which are invalid):


```
8.0.0 - 8.0.1
8.1.5 and later
<= 2
prior to 4.2
All versions < V3.0.29
V3.0, V4.0, V4.1, V4.2
```

All the examples above contain some kind of a version range and are therefore invalid under the category `product_version`.

3.1.2.3.2 Branches Type - Name under Product Version Range

If adjacent property `category` has the value `product_version_range`, the value of `name` MUST contain version ranges. The value of MUST obey to exactly one of the following options:

1. Version Range Specifier (vers)

   vers is an ongoing community effort to address the problem of version ranges. Its draft specification is available at [VERS].

   vers MUST be used in its canonical form. To convey the term "all versions", the special string `vers:all/*` MUST be used.

   According to the interpretation used here, the canonical form requires that the vers is normalized.

   Examples 1 (for `name` when using `product_version_range` with vers):

   ```
   vers:gem/>=2.2.0|!=2.2.1|<2.3.0
   vers:npm/1.2.3|>=2.0.0|<5.0.0
   vers:pypi/0.0.0|0.0.1|0.0.2|0.0.3|1.0|2.0pre1
   vers:tomee/>=8.0.0-M1|<=8.0.1
   ```

   Through the definitions of the vers specification a user can compute whether a given version is in a given range.

2. Vers-like Specifier (vls)

   This option uses only the `<version-constraint>` part from the vers specification. It MUST NOT have an URI nor the `<versioning-scheme>` part. It is a fallback option and SHOULD NOT be used unless really necessary.

   The reason for that is, that it is nearly impossible for tools to reliable determine whether a given version is in the range or not. Also, different tools might come to different results regarding this question.

   Tools MAY support this on best effort basis.

   Examples 2 (for `name` when using `product_version_range` with vls):

   ```
   <=2
   <4.2
   <V3.0.29
   >=8.1.5
   >10.9a|!=10.9c|!=10.9f|<=10.9k
   <2024-4-pabc0019|>2024-10-pefd0010|<2024-12-pjkl2010|>2024-12-pjkl5010|<=2025-
1-pghi1001
   ```


3.1.2.4 Branches Type - Product

Product (`product`) has the value type Full Product Name (`full_product_name_t`).

3.1.3 Extensions Type

List of extensions (`extensions_t`) of value type `array` with `1` or more unique items (a `set`) contains a list of extension elements for the current context.

```
# ...
extensions_t: Sequence
# ...
```

Each item MUST comply with the rules for CSAF Extensions and validate against the CSAF Extension Content schema as well as the schema given through its `$schema` property.

3.1.4 Full Product Name Type

Full Product Name (`full_product_name_t`) of value type `object` specifies information about the product and assigns the product ID. The properties `name` and `product_id` are required. The property `product_identification_helper` is optional.

```
# ...
full_product_name_t:
  name: String
  product_id: $defs.product_id_t
  product_identification_helper: Mapping
# ...
```

3.1.4.1 Full Product Name Type - Name

Textual description of the product (`name`) has value type `string` with `1` or more characters. The value SHOULD be the product's full canonical name, including version number and other attributes, as it would be used in a human-friendly document.

Examples 1:

```
Cisco AnyConnect Secure Mobility Client 2.3.185
Microsoft Host Integration Server 2006 Service Pack 1
```

3.1.4.2 Full Product Name Type - Product ID

Product ID (`product_id`) holds a value of type Product ID (`product_id_t`).

3.1.4.3 Full Product Name Type - Product Identification Helper

Helper to identify the product (`product_identification_helper`) of value type `object` provides in its properties at least one method which aids in identifying the product in an asset database. Of the given eight properties `cpe`, `hashes`, `model_numbers`, `purls`, `sbom_urls`, `serial_numbers`, `skus`, and `x_generic_uris`, `1` is mandatory.


```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    cpe: String.Pattern
    hashes: Sequence
    model_numbers: Sequence
    purls: Sequence
    sbom_urls: Sequence
    serial_numbers: Sequence
    skus: Sequence
    x_generic_uris: Sequence
# ...
```

A helper to identify the product SHALL identify the product described by the `name` in its entirety. A matching algorithm has to be able to rely on the `product_identification_helper` information during the matching without having to check e.g. the branches hierarchy leading to the product.

> Therefore, it is, for example, prohibited to omit the version from the CPE if a `product_version` was given in branches hierarchy leading to the product.

3.1.4.3.1 Full Product Name Type - Product Identification Helper - CPE

Common Platform Enumeration representation (`cpe`) has value type `string` of `5` or more characters with `pattern` (regular expression):

```
    ^((cpe:2\\.3:[aho\\*\\-](:((((\\?*|\\*?)([a-zA-Z0-9\\-\\._]|(\\\\[\\\\\\*\\?!\"#\\$%
]){5}(:(([a-zA-Z]{2,3}(-([a-zA-Z]{2}|[0-9]{3}))?)|[\\*\\-]))(:((((\\?*|\\*?)([a-zA-
Z0-9\\-\\._]|(\\\\[\\\\\\*\\?!\"#\\$%&'\\(\\)\\+,\\/:;<=>@\\[\\]\\^`\\{\\|\\}~]))+(\\?*
]){4})|([c][pP][eE]:\\/[AHOaho]?(:[A-Za-z0-9\\._\\-~%]*){0,6}))$
```

The Common Platform Enumeration (CPE) attribute refers to a method for naming platforms external to this specification. See [CPE23-N] for details.

> Both, CPE 2.2 and CPE 2.3, are supported in CSAF.

3.1.4.3.2 Full Product Name Type - Product Identification Helper - Hashes

List of hashes (`hashes`) of value type `array` holding at least `1` item contains a list of cryptographic hashes usable to identify files.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    hashes: Sequence
    # ...
# ...
```

Cryptographic hashes of value type `object` contains all information to identify a file based on its cryptographic hash values. Any cryptographic hashes object has the two mandatory properties `file_hashes` and `filename`.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    hashes:
```



```
      file_hashes: Sequence
      filename: String
    # ...
# ...
```

List of file hashes (`file_hashes`) of value type `array` holding at least `1` item contains a list of cryptographic hashes for this file.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    hashes:
      file_hashes: Sequence
      # ...
    # ...
# ...
```

Each File hash of value type `object` contains one hash value and algorithm of the file to be identified. Any File hash object has the two mandatory properties `algorithm` and `value`.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    hashes:
      file_hashes:
      - # <file_hash-instance>:
        algorithm: String.Pattern
        value: String.Pattern
      # ...
    # ...
# ...
```

The algorithm of the cryptographic hash representation (`algorithm`) has value type `string` with `1` or more characters with `pattern` (regular expression):

```
^[0-9a-z][0-9a-z-]*$
```

The algorithm of the cryptographic hash representation contains the name of the cryptographic hash algorithm used to calculate the value. The default value for `algorithm` is `sha256`. Secure cryptographic hash algorithms SHOULD be preferred.

Examples 1:

```
blake2b512
sha256
sha3-512
sha384
sha512
```

These values are derived from the currently supported digests OpenSSL [OPENSSL]. Leading dashes were removed.

The command `openssl dgst -list` (Version 3.4.0 from 2024-10-22) outputs the following:

```
Supported digests:
-blake2b512               -blake2s256               -md4
-md5                      -md5-sha1                 -mdc2
```



```
-ripemd                 -ripemd160              -rmd160
-sha1                   -sha224                 -sha256
-sha3-224               -sha3-256               -sha3-384
-sha3-512               -sha384                 -sha512
-sha512-224             -sha512-256             -shake128
-shake256               -sm3                    -ssl3-md5
-ssl3-sha1              -whirlpool
```

The Value of the cryptographic hash representation (`value`) has value type `string` of `32` or more characters with `pattern` (regular expression):

```
^[0-9a-f]{32,}$
```

The Value of the cryptographic hash attribute contains the cryptographic hash value in lowercase hexadecimal representation.

Examples 2:

```
37df33cb7464da5c7f077f4d56a32bc84987ec1d85b234537c1c1a4d4fc8d09dc29e2e762cb5203677b
4775203615d9534a8bfca96a93dc8b461a489f69124a130d786b42204f3341cc
9ea4c8200113d49d26505da0e02e2f49055dc078d1ad7a419b32e291c7afebbb84badfbd46dec42883b
```

The filename representation (`filename`) of value type `string` with `1` or more characters contains the name of the file which is identified by the hash values.

Examples 3:

```
WINWORD.EXE
msotadddin.dll
sudoers.so
```

If the value of the hash matches and the filename does not, a user SHOULD prefer the hash value. In such cases, the filename SHOULD be used as informational property.

3.1.4.3.3 Full Product Name Type - Product Identification Helper - Model Numbers

The list of models (`model_numbers`) of value type `array` with `1` or more unique items contains a list of model numbers.

A list of models SHOULD only be used if a certain range of model numbers with its corresponding software version is affected, or the model numbers change during update.

This can also be used to identify hardware. If necessary, the software, or any other related part, SHALL be bind to that via a product path.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    model_numbers: Sequence
    # ...
# ...
```

Any given model number of value type `string` with at least `1` character represents a model number of the component to identify - possibly with placeholders.

> The terms “model”, “model number” and “model variant” are mostly used synonymously. Often it is abbreviated as “MN”, M/N” or “model no.”.

If a part of a model number of the component to identify is given, it MUST begin at the first and end at the last character position of the string representing the targeted component. The wildcard characters `?` (for a single character) and `*` (for zero or more characters) signal exclusion of characters at these positions from matching. This applies also to the first character. An unescaped `*` MUST be the only `*` wildcard in the string. As part of the model number, the special characters `?`, `*` and `\` MUST be escaped with `\`.

> Note: A backslash MUST be escaped itself in a JSON string.

Examples 1:

```
*-G109A/EU?
2024-*
6RA8096-4MV62-0AA0
6RA801?-??V62-0AA0
IC25T060ATCS05-0
```

3.1.4.3.4 Full Product Name Type - Product Identification Helper - PURLs

List of PURLs (`purls`) of value type `array` with `1` or more unique items contains a list of Package-URL (PURL) identifiers.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    purls: Sequence
    # ...
# ...
```

A Package-URL representation has value type `string` of `7` or more characters with `pattern` (regular expression):

```
^pkg:[a-z][a-z0-9\\.\\-]*\\/.+
```

> The given pattern does not completely evaluate whether a PURL is valid according to the [ECMA-427] specification. It provides a more generic approach and general guidance to enable forward compatibility. CSAF uses only the canonical form of PURL to conform with section 3.3 of [RFC3986]. Therefore, URLs starting with `pkg://` are considered invalid.

The PURL attribute refers to a method for reliably identifying and locating software packages external to this specification. See [ECMA-427] and [PURL] for details. Multiple PURLs can be specified to allow for identifiers to locate identical components in different locations.

If multiple PURLs are specified, they SHALL only differ in their qualifiers. Otherwise, separate product branches SHOULD be used to differentiate between the components.

3.1.4.3.5 Full Product Name Type - Product Identification Helper - SBOM URLs

The list of SBOM URLs (`sbom_urls`) of value type `array` with `1` or more items contains a list of URLs where SBOMs for this product can be retrieved.

> The SBOMs might differ in format or depth of detail. Currently supported formats are SPDX, CycloneDX, and SWID.


```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    sbom_urls: Sequence
    # ...
# ...
```

Any given SBOM URL of value type `string` with format `uri` contains a URL of one SBOM for this product.

Examples 1:

```
https://raw.githubusercontent.com/CycloneDX/bom-examples/master/SBOM/keycloak-10.0.
https://swinslow.net/spdx-examples/example4/main-bin-v2
```

3.1.4.3.6 Full Product Name Type - Product Identification Helper - Serial Numbers

The list of serial numbers (`serial_numbers`) of value type `array` with `1` or more unique items contains a list of serial numbers.

A list of serial numbers SHOULD only be used if a certain range of serial numbers with its corresponding software version is affected, or the serial numbers change during update.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    serial_numbers: Sequence
    # ...
# ...
```

Any given serial number of value type `string` with at least `1` character represents a serial number of the component to identify - possibly with placeholders.

If a part of a serial number of the component to identify is given, it MUST begin at the first and end at the last character position of the string representing the targeted component. The wildcard characters `?` (for a single character) and `*` (for zero or more characters) signal exclusion of characters at these positions from matching. This applies also to the first character. An unescaped `*` MUST be the only `*` wildcard in the string. As part of the serial number, the special characters `?`, `*` and `\` MUST be escaped with `\`.

> Note: A backslash MUST be escaped itself in a JSON string.

Examples 1:

```
*RF8R71YR???
11S45N0249Z1ZS9*
DSEP147100
L15-VM-???
L234.696.30.044.712
```

3.1.4.3.7 Full Product Name Type - Product Identification Helper - SKUs

The list of stock keeping units (`skus`) of value type `array` with `1` or more items contains a list of stock keeping units.

A list of stock keeping units SHOULD only be used if the list of product paths is used to decouple e.g. hardware from the software, or the stock keeping units change during update. In the latter case the remediations SHALL include the new stock keeping units or a description how it can be obtained.

> The use of the list of product paths in the first case is important. Otherwise, the end user is unable to identify which version (the affected or the not affected / fixed one) is used.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    skus: Sequence
    # ...
# ...
```

Any given stock keeping unit of value type `string` with at least `1` character represents a stock keeping unit (SKU) of the component to identify - possibly with placeholders.

> Sometimes this is also called “item number”, “article number” or “product number”.

If a part of a stock keeping unit of the component to identify is given, it MUST begin at the first and end at the last character position of the string representing the targeted component. The wildcard characters `?` (for a single character) and `*` (for zero or more characters) signal exclusion of characters at these positions from matching. This applies also to the first character. An unescaped `*` MUST be the only `*` wildcard in the string. As part of the stock keeping unit, the special characters `?`, `*` and `\` MUST be escaped with `\`.

> Note: A backslash MUST be escaped itself in a JSON string.

Examples 1:

```
11????2
320196154130703
i800*
i8005-5.0
2.063.*.?:221-A
```

3.1.4.3.8 Full Product Name Type - Product Identification Helper - Generic URIs

List of generic URIs (`x_generic_uris`) of value type `array` with at least `1` item contains a list of identifiers which are either vendor-specific or derived from a standard not yet supported.

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    x_generic_uris: Sequence
# ...
```

Any such Generic URI item of value type `object` provides the two mandatory properties Namespace (`namespace`) and URI (`uri`).

```
# ...
full_product_name_t:
  # ...
  product_identification_helper:
    # ...
    x_generic_uris:
    - # <x_generic_uri-instance>:
      namespace: String.URI
      uri: String.URI
# ...
```


The namespace of the generic URI (`namespace`) of value type `string` with format `uri` refers to a URL which provides the name and knowledge about the specification used or is the namespace in which these values are valid.

The URI (`uri`) of value type `string` with format `uri` contains the identifier itself.

These elements can be used to reference a specific component from an SBOM:

Example 1 (linking a component from a CycloneDX SBOM using the bomlink mechanism):

```
          "x_generic_uris": [
            {
              "namespace": "https://cyclonedx.org/capabilities/bomlink/",
              "uri": "urn:cdx:411dafd2-c29f-491a-97d7-e97de5bc2289/1#pkg:maven/org.jbos
logging@3.4.1.Final?type=jar"
            }
          ]
```

Example 2 (linking a component from an SPDX SBOM):

```
          "x_generic_uris": [
            {
              "namespace": "https://spdx.github.io/spdx-spec/latest/document-creation-
information/#65-spdx-document-namespace-field",
              "uri": "https://swinslow.net/spdx-examples/example4/main-bin-v2#SPDXRef-
libc"
            }
          ]
```

3.1.4.4 Full Product Name Type - Extensions

Product-level Extensions (`x_extensions`) of value type Extensions Type (`extensions_t`) contains list of extensions valid at the full product name element level of the CSAF document and associated with this full product name element.

```
    "x_extensions": {
      // ...
    }
```

3.1.5 Language Type

Language type (`lang_t`) has value type `string` with `pattern` (regular expression):

```
    ^(([A-Za-z]{2,3}(-[A-Za-z]{3}(-[A-Za-z]{3}){0,2})?|[A-Za-z]{4,8})(-[A-Za-z]{4})?(-
([A-Za-z]{2}|[0-9]{3}))?(-([A-Za-z0-9]{5,8}|[0-9][A-Za-z0-9]{3}))*(-[A-WY-Za-wy-z0-
9](-[A-Za-z0-9]{2,8})+)*(-[Xx](-[A-Za-z0-9]{1,8})+)?|[Xx](-[A-Za-z0-9]{1,8})+|[Ii]-
[Dd][Ee][Ff][Aa][Uu][Ll][Tt]|[Ii]-[Mm][Ii][Nn][Gg][Oo])$
```

The value identifies a language, corresponding to IETF BCP 47 / RFC 5646. See IETF language registry: https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry

CSAF skips those grandfathered language tags that are deprecated at the time of writing the specification. Even though the private use language tags are supported they should not be used to ensure readability across the ecosystem. It is recommended to follow the conventions for the capitalization of the subtags even though it is not mandatory as

most users are used to that.

CSAF Producers SHALL follow the conventions for the capitalization of the subtags. CSAF Validators SHALL treat subtags as case-insensitive unless stated otherwise in the test.

Examples 1:

```
de
en
fr
frc
jp
```

3.1.6 Notes Type

List of notes (`notes_t`) of value type `array` with `1` or more items of type `Note` contains notes which are specific to the current context.

```
# ...
notes_t: Sequence
# ...
```

Value type of every such Note item is `object` with the mandatory properties `category` and `text` providing a place to put all manner of text blobs related to the current context. A Note `object` MAY provide the optional properties `audience`, `group_ids`, `product_ids` and `title`.

```
# ...
notes_t:
- # <note-instance>:
  audience: String
  category: String.Enum
  group_ids: $defs.product_groups_t
  product_ids: $defs.products_t
  text: String
  title: String
# ...
```

Audience of note (`audience`) of value type `string` with `1` or more characters indicates who is intended to read it.

Examples 1:

```
all
executives
operational management and system administrators
safety engineers
```

Note category (`category`) of value type `string` and `enum` contains the information of what kind of note this is. Valid `enum` values are:

```
description
details
faq
general
legal_disclaimer
other
summary
```


The value `description` indicates the note is a description of something. The optional sibling property `title` MAY have more information in this case.

The value `details` indicates the note is a low-level detailed discussion. The optional sibling property `title` MAY have more information in this case.

The value `faq` indicates the note is a list of frequently asked questions.

The value `general` indicates the note is a general, high-level note. The optional sibling property `title` MAY have more information in this case.

The value `legal_disclaimer` indicates the note represents any possible legal discussion, including constraints, surrounding the document.

The value `other` indicates the note is something that doesn't fit the other categories. The optional sibling attribute `title` SHOULD have more information to indicate clearly what kind of note to expect in this case.

The value `summary` indicates the note is a summary of something. The optional sibling property `title` MAY have more information in this case.

Group IDs (`group_ids`) are of value type Product Groups (`product_groups_t`) and contain a list of Product Groups the current note item applies to.

Product IDs (`product_ids`) are of value type Products (`products_t`) and contain a list of Products the current note item applies to.

Note content (`text`) of value type `string` with `1` or more characters holds the content of the note. Content varies depending on type.

Title of note (`title`) of value type `string` with `1` or more characters provides a concise description of what is contained in the text of the note.

Examples 2:

```
Details
Executive summary
Technical summary
Impact on safety systems
```

3.1.7 Product Group ID Type

The Product Group ID Type (`product_group_id_t`) of value type `string` with `1` or more characters is a reference token for product group instances. The value is a token required to identify a group of products so that it can be referred to from other parts in the document. There is no predefined or required format for the Product Group ID (`product_group_id`) as long as it uniquely identifies a product group in the context of the current document.

```
# ...
product_group_id_t: String
# ...
```

Examples 1:

```
CSAFGID-0001
CSAFGID-0002
CSAFGID-0020
```

Even though the standard does not require a specific format it is recommended to use different prefixes for the Product ID and the Product Group ID to support reading and parsing the document.

3.1.8 Product Groups Type

List of Product Group ID (`product_groups_t`) of value type `array` with `1` or more unique items (a `set`) of type Product Group ID (`product_group_id_t`) specifies a list of `product_group_ids` to give context to the parent item.

```
# ...
product_groups_t: Sequence
# ...
```

3.1.9 Product ID Type

The Product ID Type (`product_id_t`) of value type `string` with `1` or more characters is a reference token for product instances. The value is a token required to identify a `full_product_name` so that it can be referred to from other parts in the document. There is no predefined or required format for the Product ID (`product_id`) as long as it uniquely identifies a product in the context of the current document.

```
# ...
product_id_t: String
# ...
```

Examples 1:

```
CSAFPID-0004
CSAFPID-0008
```

Even though the standard does not require a specific format it is recommended to use different prefixes for the Product ID and the Product Group ID to support reading and parsing the document.

3.1.10 Products Type

List of Product IDs (`products_t`) of value type `array` with `1` or more unique items (a `set`) of type Product ID (`product_id_t`) specifies a list of `product_ids` to give context to the parent item.

```
# ...
products_t: Sequence
# ...
```

3.1.11 References Type

List of references (`references_t`) of value type `array` with `1` or more items of type Reference holds a list of Reference objects.

```
# ...
references_t: Sequence
# ...
```

Value type of every such Reference item is `object` with the mandatory properties `url` and `summary` holding any reference to conferences, papers, advisories, and other resources that are related and considered related to either a surrounding part of or the entire document and to be of value to the document consumer. A reference `object` MAY provide the optional property `category`.


```
# ...
references:
- # <reference-instance>:
  category: String
  summary: String
  url: String.URI
# ...
```

Category of reference (`category`) of value type `string` and `enum` indicates whether the reference points to the same document or vulnerability in focus (depending on scope) or to an external resource. Valid `enum` values are:

```
external
self
```

The default value for `category` is `external`.

The value `external` indicates, that this document is an external reference to a document or vulnerability in focus (depending on scope).

The value `self` indicates, that this document is a reference to this same document or vulnerability (also depending on scope).

> This includes links to documents with the same content but different file format (e.g. advisories as PDF or HTML).

Summary of the reference (`summary`) of value type `string` with `1` or more characters indicates what this reference refers to.

URL of reference (`url`) of value type `string` with format `uri` provides the URL for the reference.

3.1.12 Subpath Type

Subpath (`subpath_t`) of value type `object` contains the next node along the current path and its relationship to the previous node. The properties `category` and `next_product_reference` are required.

```
# ...
subpath_t:
  category: String.Enum
  next_product_reference: $defs.product_id_t
# ...
```

Relationship category (`category`) of value type `string` and `enum` defines the category of relationship between the previous item and the referenced next product. Valid `enum` values are:

```
default_component_of
external_component_of
installed_on
installed_with
optional_component_of
```

The value `default_component_of` indicates that the entity labeled with one Product ID (e.g. CSAFPID-0001) is a default component of an entity with another Product ID (e.g. CSAFPID-0002). These Product IDs SHOULD NOT be identical to provide minimal redundancy.

The value `external_component_of` indicates that the entity labeled with one Product ID (e.g. CSAFPID-0001) is an external component of an entity with another Product ID (e.g. CSAFPID-0002). These Product IDs SHOULD NOT be identical to provide minimal redundancy.

The value `installed_on` indicates that the entity labeled with one Product ID (e.g. CSAFPID-0001) is installed on a platform entity with another Product ID (e.g. CSAFPID-0002). These Product IDs SHOULD NOT be identical to provide minimal redundancy.

The value `installed_with` indicates that the entity labeled with one Product ID (e.g. CSAFPID-0001) is installed alongside an entity with another Product ID (e.g. CSAFPID-0002). These Product IDs SHOULD NOT be identical to provide minimal redundancy.

The value `optional_component_of` indicates that the entity labeled with one Product ID (e.g. CSAFPID-0001) is an optional component of an entity with another Product ID (e.g. CSAFPID-0002). These Product IDs SHOULD NOT be identical to provide minimal redundancy.

Next product reference (`next_product_reference`) of value type Product ID (`product_id_t`) holds a Product ID that refers to the Full Product Name element, which is referenced as the second element of the relationship.

3.1.13 Version Type

The Version (`version_t`) type has value type `string` with `pattern` (regular expression):

```
    ^(0|[1-9][0-9]*)$|^((0|[1-9]\\d*)\\.(0|[1-9]\\d*)\\.(0|[1-9]\\d*)(?:-((?:0|[1-9]\\d
zA-Z-][0-9a-zA-Z-]*)(?:\\.(?:0|[1-9]\\d*|\\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\\+([0-
9a-zA-Z-]+(?:\\.[0-9a-zA-Z-]+)*))?)$
```

The version specifies a version string to denote clearly the evolution of the content of the document. There are two options how it can be used:

- semantic versioning (preferred; according to the rules below)
- integer versioning

A CSAF document MUST use only one versioning system.

Examples 1:

```
1
4
0.9.0
1.4.3
2.40.0+21AF26D3
```

3.1.13.1 Version Type - Integer Versioning

Integer versioning increments for each version where the `/document/tracking/status` is `final` the version number by one. The regular expression for this type is:

```
^(0|[1-9][0-9]*)$
```

The following rules apply:

1. Once a versioned document has been released, the contents of that version MUST NOT be modified. Any modifications MUST be released as a new version.
2. Version zero (`0`) is for initial development before the `initial_release_date`. The document status MUST be `draft`. Anything MAY change at any time. The document SHOULD NOT be considered stable.
3. Version `1` defines the initial release to the specified target group. Each new version where `/document/tracking/status` is `final` has a version number incremented by one.

4. Pre-release versions (document status `draft`) MUST carry the new version number. Sole exception is before the initial release (see rule 2). The combination of document status `draft` and version `1` MAY be used to indicate that the content is unlikely to change.
5. Build metadata is never included in the version.
6. Precedence MUST be determined by integer comparison.

3.1.13.2 Version Type - Semantic Versioning

Semantic versioning derived the rules from [SemVer]. The regular expression for this type is:

```
^((0|[1-9]\\d*)\\.(0|[1-9]\\d*)\\.(0|[1-9]\\d*)(?:-((?:0|[1-9]\\d*|\\d*[a-zA-Z-][0-
9a-zA-Z-]*)(?:\\.(?:0|[1-9]\\d*|\\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\\+([0-9a-zA-Z-]+(?:
9a-zA-Z-]+)*))?)$
```

The goal of this structure is to provide additional information to the end user whether a new comparison with the asset database is needed. The "public API" in regard to CSAF is the CSAF document with its structure and content. This results in the following rules:

1. A normal version number MUST take the form `X.Y.Z` where `X`, `Y`, and `Z` are non-negative integers, and MUST NOT contain leading zeroes. `X` is the major version, `Y` is the minor version, and `Z` is the patch version. Each element MUST increase numerically. For instance: `1.9.0` -> `1.10.0` -> `1.11.0`.
2. Once a versioned document has been released, the contents of that version MUST NOT be modified. Any modifications MUST be released as a new version.
3. Major version zero (`0.y.z`) is for initial development before the `initial_release_date`. The document status MUST be `draft`. Anything MAY change at any time. The document SHOULD NOT be considered stable. Changes which would increment the major version according to rule 7 are tracked in this stage with (`0.y.z`) by incrementing the minor version `y` instead. Changes that would increment the minor or patch version according to rule 6 or 5 are both tracked in this stage with (`0.y.z`) by incrementing the patch version `z` instead.
4. Version `1.0.0` defines the initial release to the specified target group. The way in which the version number is incremented after this release is dependent on the content and structure of the document and how it changes.
5. Patch version `Z` (`x.y.Z` | `x > 0`) MUST be incremented if only backwards compatible bug fixes are introduced. A bug fix is defined as an internal change that fixes incorrect behavior.

   > In the context of the document this is the case e.g. for spelling mistakes.

6. Minor version `Y` (`x.Y.z` | `x > 0`) MUST be incremented if the content of an existing element changes except for those which are covered through rule 7. It MUST be incremented if substantial new information are introduced or new elements are provided. It MAY include patch level changes. Patch version MUST be reset to `0` when minor version is incremented.
7. Major version `X` (`X.y.z` | `X > 0`) MUST be incremented if a new comparison with the end user's asset database is required. This includes:

   - changes (adding, removing elements or modifying content) in `/product_tree` or elements which contain `/product_tree` in their path
   - adding or removing items of `/vulnerabilities`
   - adding or removing elements in:
     - `/vulnerabilities[]/product_status/first_affected`
     - `/vulnerabilities[]/product_status/known_affected`
     - `/vulnerabilities[]/product_status/last_affected`
   - removing elements from:


```
- /vulnerabilities[]/product_status/first_fixed
- /vulnerabilities[]/product_status/fixed
- /vulnerabilities[]/product_status/known_not_affected
```

It MAY also include minor and patch level changes. Patch and minor version MUST be reset to `0` when major version is incremented.

8. A pre-release version (document status `draft`) MAY be denoted by appending a hyphen and a series of dot separated identifiers immediately following the patch version. Identifiers MUST comprise only ASCII alphanumerics and hyphens `[0-9A-Za-z-]`. Identifiers MUST NOT be empty. Numeric identifiers MUST NOT include leading zeroes. Pre-release versions have a lower precedence than the associated normal version. A pre-release version indicates that the version is unstable and might not satisfy the intended compatibility requirements as denoted by its associated normal version.

   Examples 1:

   ```
   1.0.0-0.3.7
   1.0.0-alpha
   1.0.0-alpha.1
   1.0.0-x-y-z.--
   1.0.0-x.7.z.92
   ```

9. Pre-release MUST NOT be included if `/document/tracking/status` is `final`.

10. Build metadata MAY be denoted by appending a plus sign and a series of dot separated identifiers immediately following the patch or pre-release version. Identifiers MUST comprise only ASCII alphanumerics and hyphens `[0-9A-Za-z-]`. Identifiers MUST NOT be empty. Build metadata MUST be ignored when determining version precedence. Thus two versions that differ only in the build metadata, have the same precedence.

    Examples 2:

    ```
    1.0.0+20130313144700
    1.0.0+21AF26D3----117B344092BD
    1.0.0-alpha+001
    1.0.0-beta+exp.sha.5114f85
    ```

11. Precedence refers to how versions are compared to each other when ordered.

    1. Precedence MUST be calculated by separating the version into major, minor, patch and pre-release identifiers in that order (Build metadata does not figure into precedence).
    2. Precedence is determined by the first difference when comparing each of these identifiers from left to right as follows: Major, minor, and patch versions are always compared numerically.

       Example 3:

       ```
       1.0.0 < 2.0.0 < 2.1.0 < 2.1.1
       ```

    3. When major, minor, and patch are equal, a pre-release version has lower precedence than a normal version:

       Example 4:

       ```
       1.0.0-alpha < 1.0.0
       ```

    4. Precedence for two pre-release versions with the same major, minor, and patch version MUST be determined by comparing each dot separated identifier from left to right until a difference is found as follows:

       1. Identifiers consisting of only digits are compared numerically.
       2. Identifiers with letters or hyphens are compared lexically in ASCII sort order.
       3. Numeric identifiers always have lower precedence than non-numeric identifiers.

4. A larger set of pre-release fields has a higher precedence than a smaller set, if all of the preceding identifiers are equal.

Example 5:

```
1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-alpha.beta < 1.0.0-beta < 1.0.0-beta.2 < 1.
beta.11 < 1.0.0-rc.1 < 1.0.0
```

Note, that the following values do no conform the semantic versioning described above.

Examples 6 (which are invalid):

```
1.16.13.14-Cor
1.0.0-x-y-z.-
1.0.0+21AF26D3—-117B344092BD
2.5.20+3f93da6b+7cc
3.20.0-00
```

3.2 Properties

These final five subsections document the five properties of a CSAF document. The two mandatory properties `$schema` and `document`, as well as the optional properties `product_tree`, `vulnerabilities`, `x_extensions` in that order.

3.2.1 Schema Property

JSON schema (`$schema`) of value type `string` and `enum` with format `uri` contains the URL of the CSAF JSON schema which the document promises to be valid for. The single valid value for this `enum` is:

```
https://docs.oasis-open.org/csaf/csaf/v2.1/schema/csaf.json
```

This value allows for tools to identify that a JSON document is meant to be valid against this schema. Tools can use that to support users by automatically checking whether the CSAF adheres to the JSON schema identified by this URL.

3.2.2 Document Property

Document level meta-data (`document`) of value type `object` with the six mandatory properties Category (`category`), CSAF Version (`csaf_version`), Distribution (`distribution`), Publisher (`publisher`), Title (`title`), and Tracking (`tracking`) captures the meta-data about this document describing a particular set of security advisories. In addition, the `document` object MAY provide the eight optional properties Acknowledgments (`acknowledgments`), Aggregate Severity (`aggregate_severity`), Language (`lang`), License expression (`license_expression`), Notes (`notes`), References (`references`), Source Language (`source_lang`), and Document-level Extensions (`x_extensions`) .

```
document:
  acknowledgments: $defs.acknowledgments_t
  aggregate_severity: Mapping
  category: String.Pattern
  csaf_version: String.Enum
  distribution: Mapping
  lang: $defs.lang_t
  license_expression: String
  notes: $defs.notes_t
  publisher: Mapping
  references: $defs.references_t
  source_lang: $defs.lang_t
  title: String
```



```
  tracking: Mapping
  x_extensions: $defs.extensions_t
```

3.2.2.1 Document Property - Acknowledgments

Document acknowledgments (`acknowledgments`) of value type Acknowledgments Type (`acknowledgments_t`) contains a list of acknowledgment elements associated with the whole document.

```
document:
  acknowledgements:  # $defs.acknowledgments_t
  - # <acknowledgement-instance>:
    names: Sequence
    organization: String
    summary: String
    urls: Sequence
    # ...
  # ...
```

3.2.2.2 Document Property - Aggregate Severity

Aggregate severity (`aggregate_severity`) of value type `object` with the mandatory property `text` and the optional property `namespace` is a vehicle that is provided by the document producer to convey the urgency and criticality with which the one or more vulnerabilities reported should be addressed. It is a document-level metric and applied to the document as a whole — not any specific vulnerability. The range of values in this field is defined according to the document producer's policies and procedures.

```
document:
  # ...
  aggregate_severity:
    namespace: String
    text: String
  # ...
```

The Namespace of aggregate severity (`namespace`) of value type `string` with format `uri` points to the namespace so referenced.

The Text of aggregate severity (`text`) of value type `string` with `1` or more characters provides a severity which is independent of - and in addition to - any other standard metric for determining the impact or severity of a given vulnerability (such as CVSS).

Examples 1:

```
Critical
Important
Moderate
```

3.2.2.3 Document Property - Category

Document category (`category`) has value type `string` of `1` or more characters with `pattern` (regular expression):

```
^[^\\s\\-_\\.](.*[^\\s\\-_\\.])?$
```

Document category defines a short canonical name, chosen by the document producer, which will inform the end user as to the category of document.

It is directly related to the profiles defined in section 4.


```
document:
  # ...
  category:  String.Pattern
  # ...
```

Examples 1:

```
csaf_base
csaf_security_advisory
csaf_vex
Example Company Security Notice
```

3.2.2.4 Document Property - CSAF Version

CSAF version (`csaf_version`) of value type `string` and `enum` gives the version of the CSAF specification which the document was generated for. The single valid value for this `enum` is:

```
2.1
```

3.2.2.5 Document Property - Distribution

Rules for document sharing (`distribution`) of value type `object` with the mandatory property Traffic Light Protocol (TLP) (`tlp`) and the optional properties Sharing Group (`sharing_group`) and Text (`text`) describes any constraints on how this document might be shared.

```
document:
  # ...
  distribution:
    sharing_group: Mapping
    text: String
    tlp: Mapping
  # ...
```

If multiple values are present, the TLP information SHOULD be preferred as this aids in automation. The Sharing Group SHALL be interpreted as specification to the TLP information. Therefore, the Sharing Group MAY also be used to convey special TLP restrictions:

Examples 1:

```
E-ISAC members-only
Only releasable to European Energy sector
Releasable to NATO countries
```

> Note that for such restrictions the Sharing Group Name MUST exist and all participants MUST know the associated Sharing Group IDs to allow for automation.

3.2.2.5.1 Document Property - Distribution - Sharing Group

Sharing Group (`sharing_group`) of value type `object` with the mandatory property Sharing Group ID (`id`) and the optional property Sharing Group Name (`name`) contains information about the group this document is intended to be shared with.

```
document:
  # ...
  distribution:
    sharing_group:
      id: String.Pattern
      name: String
    # ...
  # ...
```

Sharing Group ID (`id`) has value type `string` with format `uuid` and `pattern` (regular expression):

```
    ^(([0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-[0-9a-f]{4}-[0-9a-f]{12})|([0]{8}-([0]{4}-
){3}[0]{12})|([f]{8}-([f]{4}-){3}[f]{12}))$
```

Sharing Group ID provides the unique ID for the sharing group. This ID is intended to be globally unique and MAY also be used by different issuing parties to share CSAF data within a closed group, e.g. during a Multi-Party Coordinated Vulnerability Disclosure case.

> Note, that participants in such cases usually differ. Therefore, it is advised to use one ID per case. Otherwise, the consequences of adding or removing parties from a case and the implications to other cases have to be considered.

The ID SHOULD NOT change throughout different CSAF documents, if the same sharing group is addressed. It MUST differ if a different sharing group is addressed.

> It is assumed that the ID is globally unique, if constructed according to the specification for UUID Version 4.

The ID SHALL be valid according to [RFC9562] and recorded in the 8-4-4-4-12 notation in lowercase. The ID SHALL be a UUID Version 4 for any closed sharing group, i.e. `TLP:GREEN` and above.

The following ID values SHOULD NOT be used unless there are technical reasons for them. Therefore, they are reserved for implementation-specific situations:

- A system MAY use the Max UUID for `TLP:CLEAR` CSAF documents.

  > For example, the system uses the UUID as an indication whether a user allowed to see the document. The security considerations from [RFC9562] should be reflected on.

- A system MAY use the Nil UUID for CSAF documents that MUST NOT be shared.

  > For example, the CSAF document is just being drafted and the accidental leakage should be prevented.

> Note, that both values do not indicate a closed sharing group.

A CSAF document with `TLP:CLEAR` SHOULD NOT contain a sharing group value and SHALL NOT contain any other value for the Sharing Group ID than Max UUID (`ffffffff-ffff-ffff-ffff-ffffffffffff`).

If an issuing party distributes multiple versions of a single CSAF document to different sharing groups, the rules for CSAF modifier (cf. section 9.1.8) regarding the generation of the value of `/document/tracking/id` SHALL be applied. This implies that usually the sharing group ID is used as a prefix to the original `/document/tracking/id`.

Sharing Group Name (`name`) of value type `string` with one or more characters contains a human-readable name for the sharing group.

The Sharing Group Name is optional and can be chosen freely by the entity establishing the sharing group. However, the following values are reserved for the conditions below:

- For the Max UUID, the value of `name` SHALL exist and be `Public`.
- For the Nil UUID, the value of `name` SHALL exist and be `No sharing allowed`.

3.2.2.5.2 Document Property - Distribution - Text

The Textual description (`text`) of value type `string` with `1` or more characters provides a textual description of additional constraints.

Examples 1:

```
Copyright 2024, Example Company, All Rights Reserved.
Distribute freely.
Share only on a need-to-know-basis only.
```

3.2.2.5.3 Document Property - Distribution - TLP

Traffic Light Protocol (TLP) (`tlp`) of value type `object` with the mandatory property Label (`label`) and the optional property URL (`url`) provides details about the TLP classification of the document.

```
document:
  # ...
  distribution:
    # ...
    tlp:
      label: String
      url: String.URI
  # ...
```

The Label of TLP (`label`) with value type `string` and `enum` provides the TLP label of the document. Valid values of the `enum` are:

```
AMBER
AMBER+STRICT
CLEAR
GREEN
RED
```

> Note: In the TLP specification, there are only four labels. The part `+STRICT` is an extension to `TLP:AMBER`. To simplify the JSON structure, avoid additional business level tests and aid in parsing, consumption and processing, it is provided as a label to be selected instead of having a separate field.

The default value for `label` is `CLEAR`.

> Note: This provides the suggested default value for anyone writing CSAF documents as the majority of those are intended to be publicly available.

The URL of TLP version (`url`) with value type `string` with format `uri` provides a URL where to find the textual description of the TLP version which is used in this document. The default value is the URL to the definition by FIRST:

```
https://www.first.org/tlp/
```

Examples 1:

```
https://www.us-cert.gov/tlp
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/TLP/merkblatt-tlp.pdf
```


3.2.2.6 Document Property - Language

Document language (`lang`) of value type Language Type (`lang_t`) identifies the language used by this document, corresponding to IETF BCP 47 / RFC 5646.

3.2.2.7 Document Property - License Expression

License expression (`license_expression`) of value type `string` with `1` or more characters contains the SPDX license expression for the CSAF document. It MUST NOT contain a license text. See Annex B of [SPDX301] for details. The `DocumentRef` part given in that ABNF MUST NOT be used in CSAF. Any SPDX license identifier not from the official SPDX license identifier list MUST contain a prefix of the form `LicenseRef-<license-inventoring-entity>-` where `<license-inventoring-entity>` is replaced with a unique name for the entity that provided the database this license identifier was found in. Unless otherwise previously established, the unique name SHOULD be a domain name. The same applies for `AdditionRef-` identifiers.

In addition, the following rules apply:

- License identification:
  1. The SPDX License List Version 3.26.0 or later MUST be consulted to identify the appropriate SPDX license identifier or construct an expression based on a SPDX license identifier. Deprecated license identifiers SHOULD NOT be used. SPDX license identifiers that were deprecated before the version listed above MUST NOT be used.

     > The list is available at https://spdx.org/licenses/. It includes also the exceptions.

  2. If the appropriate license identifier is not found in the SPDX License List or expression been possible to constructed, the license database AboutCode's "ScanCode LicenseDB" MUST be consulted as a next step. License identifiers from this database MUST use the prefix `LicenseRef-scancode-`.

     > The database is currently available at https://scancode-licensedb.aboutcode.org/.

     The construction of a license expression with such an identifier is also allowed.
  3. If the first two steps did not result in an appropriate license identifier or expression and no extant identifier was found, the issuing party SHALL create their own license identifier following the rules above. The license identifier SHOULD contain the version number of the license. The license text MUST be made available through exactly one document note using the `category` `legal_disclaimer`. The `title` MUST be `License` for English or an unspecified document language. For any other language, it SHOULD be the language specific translation of that term.
- License similarity:
  - Annex C of [SPDX301] applies.

Examples 1:

```
CC-BY-4.0
LicenseRef-www.example.org-Example-CSAF-License-3.0
LicenseRef-scancode-public-domain
MIT OR any-OSI
```

When choosing a license expression, CSAF issuing parties SHALL consider to use a permissive license to enable the flow of information through the ecosystem.

> Restrictive licenses can prevent users to easily utilize tools that are hosted in the cloud as the tool providers cannot pre-import CSAF document but users need at least to activate the sources themselves. This is relevant for services and tools, regardless of whether they are commercial or not.

3.2.2.8 Document Property - Notes

Document notes (`notes`) of value type Notes Type (`notes_t`) holds notes associated with the whole document.

```
document:
  # ...
  notes:  # $defs.notes_t
  - # <note-instance>:
    audience: String
    category: String.Enum
    group_ids: $defs.product_groups_t
    product_ids: $defs.products_t
    text: String
    title: String
  # ...
```

The following combinations of `category` and `title` have a special meaning and MUST be used as stated below:

`category`	`title`	content of `text`
`description`	Product Description	Contains a description of a product given in the `product_tree` in regard to field of application and core functionality. This SHOULD be bound to the corresponding product or product group.
`general`	General Security Recommendations	Contains general advise and security recommendations that are related, generic and might be independently applicable of the content of the CSAF document.
`legal_disclaimer`	License	Contains the only license text of the document license.
`summary`	Summary	Contains a short summary of the content of the advisory.

If a note is specific to a product or product group it MUST be bound via the `group_ids` respectively `product_ids`.

3.2.2.9 Document Property - Publisher

Publisher (`publisher`) of value type `object` with the mandatory properties Category (`category`), Name (`name`) and Namespace (`namespace`) provides information on the publishing entity. The two other optional properties are: `contact_details` and `issuing_authority`.

```
document:
  publisher:
    category: String
    contact_details: String
    issuing_authority: String
    name: String
    namespace: String
```

3.2.2.9.1 Document Property - Publisher - Category

The Category of publisher (`category`) of value type `string` and `enum` provides information about the category of publisher releasing the document. The valid values are:

```
coordinator
discoverer
multiplier
```



```
other
translator
user
vendor
```

The value `coordinator` indicates individuals or organizations that manage a single vendor's response or multiple vendors' responses to a vulnerability, a security flaw, or an incident. This includes all Computer Emergency/Incident Response Teams (CERTs/CIRTs) or agents acting on the behalf of a researcher.

The value `discoverer` indicates individuals or organizations that find vulnerabilities or security weaknesses. This includes all manner of researchers.

The value `multiplier` indicates individuals or organizations that use existing CSAF documents or information that could be represented in CSAF, and create their own CSAF documents for distribution to a specific target audience. A single multiplier might have target audiences.

> For example, a National CSIRT might create different CSAF documents for the same vulnerability for critical infrastructure companies in different sectors, government agencies, non-critical industry, and the public based on information sharing agreements and threats to the target group.

The creation step can make use of a CSAF modifier that replaces metadata, e.g. the document publisher. Currently, this value includes multipliers, republishers, and forwarders.

The value `translator` indicates individuals or organizations that translate CSAF documents. This includes all manner of language translators, also those who work for the party issuing the original advisory.

The value `other` indicates a catchall for everyone else. Currently this includes editors, reviewers, and miscellaneous contributors.

The value `user` indicates anyone using a vendor's product.

The value `vendor` indicates developers or maintainers of information system products or services. This includes all authoritative product vendors, product security incident response teams (PSIRTs), open source projects as well as product resellers and distributors, including authoritative vendor partners.

3.2.2.9.2 Document Property - Publisher - Contact Details

Contact details (`contact_details`) of value type `string` with `1` or more characters provides information on how to contact the publisher, possibly including details such as web sites, email addresses, phone numbers, and postal mail addresses.

Example 1:

```
Example Company can be reached at contact_us@example.com, or via our website at htt
```

3.2.2.9.3 Document Property - Publisher - Issuing Authority

Issuing authority (`issuing_authority`) of value type `string` with `1` or more characters Provides information about the authority of the issuing party to release the document, in particular, the party's constituency and responsibilities or other obligations.

3.2.2.9.4 Document Property - Publisher - Name

The Name of publisher (`name`) of value type `string` with `1` or more characters contains the name of the issuing party.

Examples 1:

```
BSI
Cisco PSIRT
Siemens ProductCERT
```

3.2.2.9.5 Document Property - Publisher - Namespace

The Namespace of publisher (`namespace`) of value type `string` with format `uri` contains a URL which is under control of the issuing party and can be used as a globally unique identifier for that issuing party. The URL SHALL be normalized.

An issuing party can choose any URL which fulfills the requirements state above. The URL MAY be dereferenceable. If an issuing party has chosen a URL, it SHOULD NOT change. Tools can make use of the combination of `/document/publisher/namespace` and `/document/tracking/id` as it identifies a CSAF document globally unique.

If an issuing party decides to change its Namespace it SHOULD reissue all CSAF documents with an incremented (patch) version which has no other changes than:

- the new publisher information
- the updated revision history
- the updated item in `/document/references[]` which points to the new version of the CSAF document
- an added item in `/document/references[]` which points to the previous version of the CSAF document (if the URL changed)

Examples 1:

```
https://csaf.io
https://www.example.com
```

3.2.2.10 Document Property - References

Document references (`references`) of value type References Type (`references_t`) holds a list of references associated with the whole document.

```
document:
  # ...
  references:  # $defs.references_t
  - # <reference-instance>:
    category: String
    summary: String
    url: String.URI
  # ...
```

3.2.2.11 Document Property - Source Language

Source language (`source_lang`) of value type Language Type (`lang_t`) identifies if this copy of the document is a translation then the value of this property describes from which language this document was translated.

The property MUST be present for any CSAF document with the value `translator` in `/document/publisher/category`. The property SHALL NOT be present if the document was not translated.

> If an issuing party publishes a CSAF document with the same content in more than one language, one of these documents SHOULD be deemed the “original”, the other ones SHOULD be considered translations from the “original”.

The issuing party can retain its original publisher information including the `category`. However, other rules defined in the conformance clause “CSAF translator” SHOULD be applied.

3.2.2.12 Document Property - Title

Title of this document (`title`) of value type `string` with `1` or more characters SHOULD be a canonical name for the document, and sufficiently unique to distinguish it from similar documents.

Examples 1:

```
Cisco IPv6 Crafted Packet Denial of Service Vulnerability
Example Company Cross-Site-Scripting Vulnerability in Example Generator
```

3.2.2.13 Document Property - Tracking

Tracking (`tracking`) of value type `object` with the six mandatory properties Current Release Date (`current_release_date`), Identifier (`id`), Initial Release Date (`initial_release_date`), Revision History (`revision_history`), Status (`status`), and Version (`version`) is a container designated to hold all management attributes necessary to track a CSAF document as a whole. The two optional additional properties are Aliases (`aliases`) and Generator (`generator`).

```
document:
  # ...
  tracking:
    aliases: Sequence
    current_release_date: String.DateTime
    generator: Mapping
    id: String.Pattern
    initial_release_date: String.DateTime
    revision_history: Sequence
    status: String.Enum
    version: $defs.version_t
```

3.2.2.13.1 Document Property - Tracking - Aliases

Aliases (`aliases`) of value type `array` with `1` or more unique items (a `set`) representing Alternate Names contains a list of alternate names for the same document.

```
document:
  # ...
  tracking:
    aliases: Sequence
    # ...
```

Every such Alternate Name of value type `string` with `1` or more characters specifies a non-empty string that represents a distinct optional alternative ID used to refer to the document.

Example 1:

```
CVE-2019-12345
```

3.2.2.13.2 Document Property - Tracking - Current Release Date

Current release date (`current_release_date`) of value type `string` with format `date-time` holds the date when the current revision of this document was released.

3.2.2.13.3 Document Property - Tracking - Generator

Document Generator (`generator`) of value type `object` with mandatory property Engine (`engine`) and optional property Date (`date`) is a container to hold all elements related to the generation of the document. These items will reference when the document was actually created, including the date it was generated and the entity that generated it.

```
document:
  tracking:
    # ...
    generator:
      date: String.DateTime
      engine: Mapping
    # ...
```

Date of document generation (`date`) of value type `string` with format `date-time` SHOULD be the current date that the document was generated. Because documents are often generated internally by a document producer and exist for a nonzero amount of time before being released, this field MAY be different from the Initial Release Date and Current Release Date.

Engine of document generation (`engine`) of value type `object` with mandatory property Engine name (`name`) and optional property Engine version (`version`) contains information about the engine that generated the CSAF document.

```
document:
  tracking:
    # ...
    generator:
      # ...
      engine:
        name: String
        version: String
    # ...
```

Engine name (`name`) of value type `string` with `1` or more characters represents the name of the engine that generated the CSAF document.

Examples 1:

```
Red Hat rhsa-to-cvrf
Secvisogram
TVCE
```

Engine version (`version`) of value type `string` with `1` or more characters contains the version of the engine that generated the CSAF document.

> Although it is not formally required, the TC suggests to use a versioning which is compatible with Semantic Versioning as described in the external specification [SemVer]. This could help the end user to identify when CSAF consumers have to be updated.

Examples 2:

```
0.6.0
1.0.0-beta+exp.sha.a1c44f85
2
```


3.2.2.13.4 Document Property - Tracking - ID

Unique identifier for the document (`id`) has value type `string` of `1` or more characters with `pattern` (regular expression):

```
^[\\S](.*[\\S])?$
```

Unique identifier for the document holds the Identifier. It SHALL NOT start or end with a white space and SHALL NOT contain a newline sequence.

The ID is a simple label that provides for a wide range of numbering values, types, and schemes. Its value SHOULD be assigned and maintained by the original document issuing authority. It MUST be unique for that organization.

Examples 1:

```
Example Company - 2019-YH3234
RHBA-2019:0024
cisco-sa-20190513-secureboot
```

This value is also used to determine the filename for the CSAF document (cf. section 5.1).

> The combination of `/document/publisher/namespace` and `/document/tracking/id` identifies a CSAF document globally unique. The combination of `/document/publisher/namespace`, `/document/tracking/id`, and `/document/tracking/version` identifies that specific version of the CSAF document globally unique.

3.2.2.13.5 Document Property - Tracking - Initial Release Date

Initial release date (`initial_release_date`) of value type `string` with format `date-time` holds the date when this document was first released to the specified target group.

> For `TLP:CLEAR` documents, this is usually the timestamp when the document was published. For `TLP:GREEN` and higher, this is the timestamp when it was first made available to the specific group. Note that the initial release date does not change after the initial release even if the document is later on released to a broader audience.

If the timestamp of the initial release date was set incorrectly, it MUST be corrected. This change MUST be tracked with a new entry in the revision history.

3.2.2.13.6 Document Property - Tracking - Revision History

The Revision History (`revision_history`) with value type `array` of `1` or more Revision History Entries holds one revision item for each version of the CSAF document, including the initial one.

```
document:
  # ...
  tracking:
    # ...
    revision_history: Sequence
    # ...
```

Each Revision contains all the information elements required to track the evolution of a CSAF document. Revision History Entry items are of value type `object` with the three mandatory properties: Date (`date`), Number (`number`), and Summary (`summary`). In addition, a Revision MAY expose the optional property `legacy_version`.


```
document:
  # ...
  tracking:
    # ...
    revision_history:
    - # <revision-instance>:
      date: String.DateTime
      legacy_version: String
      number: $defs.version_t
      summary: String
    # ...
```

The Date of the revision (`date`) of value type `string` with format `date-time` states the date of the revision entry.

Legacy version of the revision (`legacy_version`) of value type `string` with `1` or more characters contains the version string used in an existing document with the same content.

> This SHOULD be used to aid in the mapping between existing (human-readable) documents which might use a different version scheme and CSAF documents with the same content. It is recommended, to use the CSAF revision number to describe the revision history for any new human-readable equivalent.

The Number (`number`) has value type Version (`version_t`).

The Summary of the revision (`summary`) of value type `string` with `1` or more characters holds a single non-empty string representing a short description of the changes.

Each Revision item which has a `number` of `0` or `0.y.z` MUST be removed from the document if the document status is `final`. Versions of the document which are pre-release SHALL NOT have its own revision item. All changes MUST be tracked in the item for the next release version. Build metadata SHOULD NOT be included in the `number` of any revision item.

Any issuing party using a CSAF document as basis and modifying it MUST create a new revision history tracking the modified document. This applies especially to CSAF multiplier.

> The new revision history (and consequently the corresponding ensures `current_release_date`) that the CSAF document can be found and downloaded by CSAF downloaders that retrieve just those CSAF documents that have a newer `current_release_date` than the timestamp of their last visit at this source (incremental download).

3.2.2.13.7 Document Property - Tracking - Status

Document status (`status`) of value type `string` and `enum` defines the draft status of the document. The value MUST be one of the following:

```
draft
final
interim
```

The value `draft` indicates, that this is a pre-release, intended for issuing party's internal use only, or possibly used externally when the party is seeking feedback or indicating its intentions regarding a specific issue.

The value `final` indicates, that the issuing party asserts the content is unlikely to change. "Final" status is an indication only, and does not preclude updates. This SHOULD be used if the issuing party expects no, slow or few changes.

The value `interim` indicates, that the issuing party expects rapid updates. This SHOULD be used if the expected rate of release for this document is significant higher than for other documents. Once the rate slows down it MUST be changed to `final`. This MAY be done in a patch version.

This is extremely useful for downstream vendors to constantly inform the end users about ongoing investigation. It can be used as an indication to pull the CSAF document more frequently.

3.2.2.13.8 Document Property - Tracking - Version

Version has the value type Version (`version_t`).

3.2.2.14 Document Property - Extensions

Document-level Extensions (`x_extensions`) of value type Extensions Type (`extensions_t`) contains list of extensions valid at the document property level of the CSAF document and associated with this document metadata.

```
document:
  # ...
  x_extensions:  # $defs.extensions_t
  - # <x_extension-instance>:
    $schema: String
    category: String.Enum
    content: Mapping
    critical: Boolean
    # ...
```

3.2.3 Product Tree Property

Product Tree (`product_tree`) has value type `object` with `1` or more properties is a container for all fully qualified product names that can be referenced elsewhere in the document. The properties are Branches (`branches`), Full Product Names (`full_product_names`), Product Groups (`product_groups`), and Product Paths (`product_paths`).

```
# ...
product_tree:
  branches: $defs.branches_t
  full_product_names: Mapping
  product_groups: Sequence
  product_paths: Sequence
```

3.2.3.1 Product Tree Property - Branches

List of branches (`branches`) has the value type `branches_t`.

3.2.3.2 Product Tree Property - Full Product Names

List of full product names (`full_product_names`) of value type `array` with `1` or more items of type `full_product_name_t` contains a list of full product names.

3.2.3.3 Product Tree Property - Product Groups

List of product groups (`product_groups`) of value type `array` with `1` or more items of value type `object` contains a list of product groups.

```
# ...
product_tree:
  # ...
  product_groups: Sequence
# ...
```

The product group items are of value type `object` with the two mandatory properties Group ID (`group_id`) and Product IDs (`product_ids`) and the optional Summary (`summary`) property.


```
# ...
product_tree:
  # ...
  product_groups:
  - # <product_group-instance>:
    group_id: $defs.product_group_id_t
    product_ids: $defs.product_id_t
    summary: String
  # ...
# ...
```

The summary of the product group (`summary`) of value type `string` with `1` or more characters gives a short, optional description of the group.

Examples 1:

```
Products supporting Modbus.
The x64 versions of the operating system.
```

Group ID (`group_id`) has value type Product Group ID (`product_group_id_t`).

List of Product IDs (`product_ids`) of value type `array` with `2` or more unique items of value type Product ID (`product_id_t`) lists the product_ids of those products which known as one group in the document.

3.2.3.4 Product Tree Property - Product Paths

List of product paths (`product_paths`) of value type `array` with `1` or more items contains a list of product paths.

```
# ...
product_tree:
  # ...
  product_paths: Sequence
# ...
```

The Product path item is of value type `object` and has three mandatory properties: Beginning product reference (`beginning_product_reference`), Full Product Name (`full_product_name`), and Subpaths (`subpaths`). The Product path item establishes a path along existing full_product_name_t elements, allowing the document producer to define a path of multiple products that form a new full_product_name entry.

```
# ...
product_tree:
  # ...
  product_paths:
  - # <product_path-instance>:
    beginning_product_reference: $defs.product_id_t
    full_product_name: $defs.full_product_name_t
    subpaths: Sequence
  # ...
# ...
```

> The situation where a need for declaring a Product path arises, is given when a product is e.g. vulnerable only when installed together with another, or to describe operating system components.

Beginning product reference (`beginning_product_reference`) of value type Product ID (`product_id_t`) holds a Product ID that refers to the Full Product Name element, which is the beginning node of the product path. It is also the product, the first node in `subpaths` links to.

> The product referenced in `beginning_product_reference` is usually the product that is more closely identified by the product path.

Full Product Name (`full_product_name`) of value type Full Product Name Type (`full_product_name_t`).

List of product subpaths (`subpaths`) of value type `array` with `1` or more items of type Subpath (`subpath_t`) contains an ordered list of product subpaths, each one relating to the path defined by all previous elements up to the beginning node of the product path.

```
# ...
product_tree:
  # ...
  product_paths:
  - # <product_path-instance>:
    # ...
    subpaths:
    - # <subpath-instance>: $defs.subpath_t
      category: String.Enum
      next_product_reference: $defs.product_id_t
    # ...
  # ...
# ...
```

Example 1:

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-908070601",
      "name": "Cisco AnyConnect Secure Mobility Client 4.9.04053"
    },
    {
      "product_id": "CSAFPID-908070602",
      "name": "Microsoft Windows"
    }
  ],
  "product_paths": [
    {
      "beginning_product_reference": "CSAFPID-908070601",
      "full_product_name": {
        "product_id": "CSAFPID-908070603",
        "name": "Cisco AnyConnect Secure Mobility Client 4.9.04053 installed on Micro
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070602",
        }
      ]
    }
  ]
}
```

The product `Cisco AnyConnect Secure Mobility Client 4.9.04053"` (Product ID: `CSAFPID-908070601`) and the product `Microsoft Windows` (Product ID: `CSAFPID-908070602`) form together a new product with the separate Product ID `CSAFPID-908070603`. The latter one can be used to refer to that combination in other parts of the CSAF document. In the preceding example, it might be the case that `Cisco AnyConnect Secure Mobility Client 4.9.04053"` is only vulnerable when installed on `Microsoft Windows`.

3.2.4 Vulnerabilities Property

Vulnerabilities (`vulnerabilities`) of value type `array` with `1` or more objects representing vulnerabilities by providing one or more properties represents a list of all relevant vulnerability information items.

```
# ...
vulnerabilities: Sequence
```

The Vulnerability item of value type `object` with `1` or more properties is a container for the aggregation of all fields that are related to a single vulnerability in the document. Any vulnerability MAY provide the optional properties Acknowledgments (`acknowledgments`), Common Vulnerabilities and Exposures (CVE) (`cve`), Common Weakness Enumeration (CWE) (`cwes`), Disclosure Date (`disclosure_date`), Discovery Date (`discovery_date`), List of first known exploitation dates (`first_known_exploitation_dates`), Flags (`flags`), IDs (`ids`), Involvements (`involvements`), Metrics (`metrics`), Notes (`notes`), Product Status (`product_status`), References (`references`), Remediations (`remediations`), Threats (`threats`), Title (`title`), and Vulnerability-level Extensions (`x_extensions`).

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  acknowledgments: $defs.acknowledgments_t
  cve: String.Pattern
  cwes: Sequence
  disclosure_date: String.DateTime
  discovery_date: String.DateTime
  first_known_exploitation_dates: Sequence
  flags: Sequence
  ids: Sequence
  involvements: Sequence
  metrics: Sequence
  notes: $defs.notes_t
  product_status: Mapping
  references: $defs.references_t
  remediations: Sequence
  threats: Sequence
  title: String
  x_extensions: $defs.extensions_t
# ...
```

3.2.4.1 Vulnerabilities Property - Acknowledgments

Vulnerability acknowledgments (`acknowledgments`) of value type Acknowledgments Type (`acknowledgments_t`) contains a list of acknowledgment elements associated with this vulnerability item.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  acknowledgments:  # $defs.acknowledgments_t
  - # <acknowledgment-instance>:
    names: Sequence
    organization: String
    summary: String
    urls: Sequence
    # ...
  # ...
```

3.2.4.2 Vulnerabilities Property - CVE

CVE (`cve`) has value type `string` with `pattern` (regular expression):

```
^CVE-[0-9]{4}-[0-9]{4,}$
```


CVE holds the MITRE standard Common Vulnerabilities and Exposures (CVE) tracking number for the vulnerability.

3.2.4.3 Vulnerabilities Property - CWEs

List of CWEs (`cwes`) of value type `array` with `1` or more unique items (a set) of value type `object` contains a list of CWEs.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  cwes: Sequence
  # ...
```

If more than one CWE is specified, the most applicable weakness ID SHOULD be listed first. A CSAF viewer MAY decide to display just the first CWE item.

Every CWE item of value type `object` with the three mandatory properties Weakness ID (`id`), Weakness Name (`name`), CWE version (`version`) holds the MITRE standard Common Weakness Enumeration (CWE) for the weakness associated. For more information cf. [CWE].

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  cwes:
  - # <cwe-instance>:
    id: String.Pattern
    name: String.Pattern
    version: String.Pattern
  # ...
```

The Weakness ID (`id`) has value type `string` with `pattern` (regular expression):

```
^CWE-[1-9]\\d{0,5}$
```

It holds the ID for the weakness associated.

Examples 1:

```
CWE-22
CWE-352
CWE-79
```

The Weakness name (`name`) has value type `string` of `1` or more characters with `pattern` (regular expression):

```
^[^\\s\\-_\\.](.*[^\\s\\-_\\.])?$
```

The Weakness name holds the full name of the weakness as given in the CWE specification.

Examples 2:

```
Cross-Site Request Forgery (CSRF)
Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'
```


The CWE version (`version`) has value type `string` with `pattern` (regular expression):

```
^[1-9]\\d*\\.([0-9]|([1-9]\\d+))(\\.\\d+)?$
```

It holds the version string of the CWE specification this weakness was extracted from. When creating or modifying a CSAF document, the latest published version of the CWE specification SHOULD be used.

Examples 3:

```
"1.0",
"3.4.1",
"4.0",
"4.11",
"4.12"
```

3.2.4.4 Vulnerabilities Property - Disclosure Date

Disclosure date (`disclosure_date`) of value type `string` with format `date-time` holds the date and time the vulnerability was originally disclosed to the public.

For vulnerabilities not yet disclosed to the public, a disclosure date in the future SHOULD indicate the intended date for disclosure of the vulnerability. This is also sometimes called embargo date. As disclosure dates may change during a vulnerability disclosure process, an issuing party SHOULD produce an updated CSAF document to confirm that the vulnerability was in fact disclosed to the public at that time or update the `disclosure_date` with the new intended date in the future.

3.2.4.5 Vulnerabilities Property - Discovery Date

Discovery date (`discovery_date`) of value type `string` with format `date-time` holds the date and time the vulnerability was originally discovered.

3.2.4.6 Vulnerabilities Property - First Known Exploitation Dates

List of first known exploitation dates (`first_known_exploitation_dates`) of value type `array` with `1` or more unique items (a set) contains a list of dates of first known exploitations.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  first_known_exploitation_dates: Sequence
  # ...
```

Every First known exploitation date item of value type `object` with the two mandatory properties Date of the information (`date`) and Date of the exploitation (`exploitation_date`) holds at least `3` properties and contains information on when this vulnerability was first known to be exploited in the wild in the products specified. At least one of the optional elements Group IDs (`group_ids`) and Product IDs (`product_ids`) MUST be present to state for which products or product groups this date is applicable.

> This information can be helpful to determine the risk of compromise. It can also be used to provide an indication for the time frame to be considered in a threat hunt for the exploitation this vulnerability.


```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  first_known_exploitation_dates:
  - # <event-instance>:
    date: String.DateTime
    exploitation_date: String.DateTime
    group_ids: $defs.product_groups_t
    product_ids: $defs.products_t
  # ...
```

Date of the information (`date`) of value type `string` with format `date-time` contains the date when the information was last updated.

Date of the exploitation (`exploitation_date`) of value type `string` with format `date-time` contains the date when the exploitation happened.

> Different document issuers might have different knowledge about exploitations in the wild that happened. Therefore, the `exploitation_date` can differ.

Group IDs (`group_ids`) are of value type Product Groups (`product_groups_t`) and contain a list of Product Groups the current first known exploitation date item applies to.

Product IDs (`product_ids`) are of value type Products (`products_t`) and contain a list of Products the current first known exploitation date item applies to.

3.2.4.7 Vulnerabilities Property - Flags

List of flags (`flags`) of value type `array` with `1` or more unique items (a set) of value type `object` contains a list of machine readable flags.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  flags: Sequence
  # ...
```

Every Flag item of value type `object` with the mandatory property Label (`label`) contains product specific information in regard to this vulnerability as a single machine readable flag. For example, this could be a machine readable justification code why a product is not affected. At least one of the optional elements Group IDs (`group_ids`) and Product IDs (`product_ids`) MUST be present to state for which products or product groups this flag is applicable.

> These flags enable the receiving party to automate the selection of actions to take.

In addition, any Flag item MAY provide the three optional properties Date (`date`), Group IDs (`group_ids`) and Product IDs (`product_ids`).

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  flags:
  - # <flag-instance>:
    date: String.DateTime
    group_ids: $defs.product_groups_t
    label: String.Enum
    product_ids: $defs.products_t
  # ...
```


Date of the flag (`date`) of value type `string` with format `date-time` contains the date when assessment was done or the flag was assigned.

Group IDs (`group_ids`) are of value type Product Groups (`product_groups_t`) and contain a list of Product Groups the current flag item applies to.

Label of the flag (`label`) of value type `string` and `enum` specifies the machine readable label. Valid `enum` values are:

```
component_not_present
inline_mitigations_already_exist
vulnerable_code_cannot_be_controlled_by_adversary
vulnerable_code_not_in_execute_path
vulnerable_code_not_present
```

The given values reflect the VEX not affected justifications. See [VEX-Justification] for more details. The values MUST be used as follows:

- `component_not_present`: The software is not affected because the vulnerable component is not in the product.
- `vulnerable_code_not_present`: The product is not affected because the code underlying the vulnerability is not present in the product.

  > Unlike `component_not_present`, the component in question is present, but for whatever reason (e.g. compiler options) the specific code causing the vulnerability is not present in the component.

- `vulnerable_code_cannot_be_controlled_by_adversary`: The vulnerable component is present, and the component contains the vulnerable code. However, vulnerable code is used in such a way that an attacker cannot mount any anticipated attack.
- `vulnerable_code_not_in_execute_path`: The affected code is not reachable through the execution of the code, including non-anticipated states of the product.

  > Components that are neither used nor executed by the product.

- `inline_mitigations_already_exist`: Built-in inline controls or mitigations prevent an adversary from leveraging the vulnerability.

Product IDs (`product_ids`) are of value type Products (`products_t`) and contain a list of Products the current flag item applies to.

3.2.4.8 Vulnerabilities Property - IDs

List of IDs (`ids`) of value type `array` with `1` or more unique ID items of value type `object` represents a list of unique labels or tracking IDs for the vulnerability (if such information exists).

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  ids: Sequence
  # ...
```

Every ID item of value type `object` with the two mandatory properties System Name (`system_name`) and Text (`text`) contains a single unique label or tracking ID for the vulnerability.


```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  ids:
  - # <id-instance>:
    system_name: String
    text: String
  # ...
```

System name (`system_name`) of value type `string` with `1` or more characters indicates the name of the vulnerability tracking or numbering system.

Examples 1:

```
Cisco Bug ID
GitHub Issue
https://github.com/oasis-tcs/csaf
```

Text (`text`) of value type `string` with `1` or more characters is unique label or tracking ID for the vulnerability (if such information exists).

Examples 2:

```
CSCso66472
oasis-tcs/csaf#210
#1217
```

General examples may include an identifier from a vulnerability tracking system that is available to customers, such as:

- a Cisco bug ID,
- a GitHub Issue number,
- an ID from a Bugzilla system, or
- an ID from a public vulnerability database such as the X-Force Database.

The list of registered vulnerability id systems is available via [RVISC].
The ID MAY be a vendor-specific value but is not to be used to publish the CVE tracking numbers (MITRE standard Common Vulnerabilities and Exposures), as these are specified inside the dedicated CVE element.

3.2.4.9 Vulnerabilities Property - Involvements

List of involvements (`involvements`) of value type `array` with `1` or more unique items (a set) of value type `object` contains a list of involvements.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  involvements: Sequence
  # ...
```

Every Involvement item of value type `object` with the two mandatory properties Party (`party`), Status (`status`) and the five optional properties Party contact information (`contact`), Date of involvement (`date`), Group IDs (`group_ids`), Product IDs (`product_ids`), and Summary (`summary`) is a container that allows the document producers to comment on the level of involvement (or engagement) of themselves (or third parties) in the vulnerability identification, scoping, and remediation process. It can also be used to convey the disclosure timeline. The ordered tuple of the values of `party` and `date` (if present) SHALL be unique within `involvements`.


```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  involvements:
  - # <involvement-instance>:
    contact: String
    date: String.DateTime
    group_ids: $defs.product_groups_t
    party: String.Enum
    product_ids: $defs.products_t
    status: String.Enum
    summary: String
  # ...
```

Party contact information (`contact`) contains the contact information of the party that was used in this state.

> In many cases, that could be an email address.

Date of involvement (`date`) of value type `string` with format `date-time` holds the date and time of the involvement entry.

Group IDs (`group_ids`) are of value type Product Groups (`product_groups_t`) and contain a list of Product Groups the current involvement item applies to.

Party category (`party`) of value type `string` and `enum` defines the category of the involved party. Valid values are:

```
coordinator
discoverer
other
user
vendor
```

These values follow the same definitions as given for the publisher category (cf. section 3.2.2.9.1).

Product IDs (`product_ids`) are of value type Products (`products_t`) and contain a list of Products the current involvement item applies to.

Party status (`status`) of value type `string` and `enum` defines contact status of the involved party. Valid values are:

```
completed
contact_attempted
disputed
in_progress
not_contacted
open
```

Each status is mutually exclusive - only one status is valid for a particular vulnerability at a particular time. As the vulnerability ages, a party's involvement could move from state to state. However, in many cases, a document producer may choose not to issue CSAF documents at each state, or simply omit this element altogether. It is recommended, however, that vendors that issue CSAF documents indicating an open or in-progress involvement SHOULD eventually expect to issue a document containing one of the statuses `disputed` or `completed` as the latest one.

> The two vulnerability involvement status states, `contact_attempted` and `not_contacted` are intended for use by document producers other than vendors (such as research or coordinating entities).

The value `completed` indicates that the party asserts that investigation of the vulnerability is complete. No additional information, fixes, or documentation from the party about the vulnerability should be expected to be released.

The value `contact_attempted` indicates that the document producer attempted to contact the party.

The value `disputed` indicates that the party disputes the vulnerability report in its entirety. This status SHOULD be used when the party believes that a vulnerability report regarding a product is completely inaccurate (that there is no real underlying security vulnerability) or that the technical issue being reported has no security implications.

The value `in_progress` indicates that some hotfixes, permanent fixes, mitigations, workarounds, or patches may have been made available by the party, but more information or fixes may be released in the future. The use of this status by a vendor indicates that future information from the vendor about the vulnerability is to be expected.

The value `not_contacted` indicates that the document producer has not attempted to make contact with the party.

The value `open` is the default status. It doesn't indicate anything about the vulnerability remediation effort other than the fact that the party has acknowledged awareness of the vulnerability report. The use of this status by a vendor indicates that future updates from the vendor about the vulnerability are to be expected.

Summary of involvement (`summary`) of value type `string` with `1` or more characters contains additional context regarding what is going on.

3.2.4.10 Vulnerabilities Property - Metrics

List of metrics (`metrics`) of value type `array` with `1` or more unique items (a set) contains metric objects for the current vulnerability.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  metrics: Sequence
  # ...
```

Every Metric item of value type `object` with the mandatory properties `content` and `products` and the optional property `source` contains all metadata about the metric including products it applies to and the source and the content itself.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  metrics:
  - # <metric-instance>:
    content: Mapping
    products: $defs.products_t
    source: String.URI
  # ...
```

3.2.4.10.1 Vulnerabilities Property - Metrics - Content

Content (`content`) of value type `object` with the optional properties CVSS v2 (`cvss_v2`), CVSS v3 (`cvss_v3`), CVSS v4 (`cvss_v4`), EPSS (`epss`), Qualitative Severity Rating (`qualitative_severity_rating`), SSVC v2 (`ssvc_v2`), and Metrics-content-level Extensions (`x_extensions`) specifies information about (at least one) metric or score for the given products regarding the current vulnerability. A Content object has at least `1` property.

```
yaml# &CERTCC-SSVC https://certcc.github.io/SSVC/data/schema/v2/ # &FIRST-CVSS https://
<csaf-instance>:   # ...   vulnerabilities:   - # <vulnerability-instance>:     # ...
metrics:     - # <metric-instance>:       content:         cvss_v2: $ref.eval(concat(
*FIRST-CVSS 'cvss-v2.0.json' ))         cvss_v3:          # !OneOf<           - $ref.e
```



```
*FIRST-CVSS 'cvss-v3.0.json' ))           - $ref.eval(concat( *FIRST-CVSS 'cvss-v3.1.js
))           # >         cvss_v4: $ref.eval(concat( *FIRST-CVSS 'cvss-v4.0.1.json' ))
epss: Mapping         qualitative_severity_rating: String.Enum         ssvc_v2:
*CERTCC-SSVC 'Decision_Point_Value_Selection-2-0-0.schema.json'           ))         x_
$defs.extensions_t       # ...     # ...
```

The property CVSS v2 (`cvss_v2`) holding a CVSS v2.0 value abiding by the schema at https://www.first.org/cvss/cvss-v2.0.json. See [CVSS2] for details.

The property CVSS v3 (`cvss_v3`) holding a CVSS v3.x value abiding by one of the schemas at https://www.first.org/cvss/cvss-v3.0.json or https://www.first.org/cvss/cvss-v3.1.json. See [CVSS30] respectively [CVSS31] for details.

The property CVSS v4 (`cvss_v4`) holding a CVSS v4.0 value abiding by the schema at https://www.first.org/cvss/cvss-v4.0.2.json. See [CVSS40] for details.

The property EPSS (`epss`) of value type `object` with the three mandatory properties Percentile (`percentile`), Probability (`probability`) and EPSS timestamp (`timestamp`) contains the EPSS data. See [EPSS] for details.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  metrics:
  - # <metric-instance>:
    content:
      # ...
      epss:
        percentile: String.Pattern
        probability: String.Pattern
        timestamp: String.DateTime
      # ...
    # ...
  # ...
```

Percentile (`percentile`) has value type `string` with `pattern` (regular expression):

```
^(([0]\\.([0-9])+)|([1]\\.[0]+))$
```

The value contains the rank ordering of probabilities from highest to lowest.

Probability (`probability`) has value type `string` with `pattern` (regular expression):

```
^(([0]\\.([0-9])+)|([1]\\.[0]+))$
```

The value contains the likelihood that any exploitation activity for this Vulnerability is being observed in the 30 days following the given timestamp.

EPSS timestamp (`timestamp`) of value type `string` with format `date-time` holds the date and time the EPSS value was recorded.

The property Qualitative Severity Rating (`qualitative_severity_rating`) of value type `string` and `enum` contains an assessment of the severity of the vulnerability regarding the products on a qualitative scale. Valid `enum` values are:

```
"critical",
"high",
"low",
"medium",
"none"
```


The value `critical` indicates that this vulnerability allows attackers to fully compromise a system or access sensitive data with little or no user interaction.

The value `high` indicates that this vulnerability can lead to significant impact such as unauthorized access or data loss, but usually require some conditions or user interaction.

The value `low` indicates that this vulnerability has a minimal impact and low likelihood of exploitation, usually causing minor issues or informational leaks.

The value `medium` indicates that this vulnerability can result in moderate impact like partial data exposure or denial of service, often needing specific circumstances.

The value `none` indicates that this flaw pose no security risk or impact like false positives or informational findings without real threat.

> The Qualitative Severity Rating is not a replacement for CVSS. It is intended to be used for sources that provide an assessment on a qualitative scale but no full CVSS vector.

CSAF consumer SHOULD give preference to CVSS if both, Qualitative Severity Rating and CVSS, are present. Issuing parties SHOULD consider using the SSVC decision point `Provider Urgency` from the `cvss` namespace to convey an additional assessment provided by a party.

The property SSVC v2 (`ssvc_v2`) holding an SSVC Selection List v2.0.0 value abiding by the schema at https://certcc.github.io/SSVC/data/schema/v2/SelectionList_2_0_0.schema.json. See [SSVC] for details.

The property Metrics-content-level Extensions (`x_extensions`) of value type Extensions Type (`extensions_t`) contains a list of extensions valid at the metrics-content-level of the CSAF document and associated with this metric element.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  metrics:
  - # <metric-instance>:
    content:
      # ...
      x_extensions:  # $defs.extensions_t
      - # <x_extension-instance>:
        $schema: String
        category: String.Enum
        content: Mapping
        critical: Boolean
        # ...
    # ...
  # ...
# ...
```

> This extension point can be used for metrics that are not supported in the CSAF standard (yet). For new versions of an official supported standard, the OASIS CSAF TC will consider providing an official extension to ensure interoperability.

3.2.4.10.2 Vulnerabilities Property - Metrics - Products

Product IDs (`products`) of value type `products_t` with `1` or more items indicates for which products the given content applies. A metric object SHOULD reflect the associated product's status (for example, a fixed product no longer contains a vulnerability and should have a CVSS score of `0`, or simply no score listed; the known affected versions of that product can list the vulnerability score as it applies to them).

3.2.4.10.3 Vulnerabilities Property - Metrics - Source

Source (`source`) of value type `string` with format `uri` contains the URL of the source that originally determined the metric. If no source is given, then the metric was assigned by the document author.

> For example, this could point to the vendor advisory, discoverer blog post, a multiplier's assessment or other sources that provide metric information.

3.2.4.11 Vulnerabilities Property - Notes

Vulnerability notes (`notes`) of value type Notes Type (`notes_t`) holds notes associated with this vulnerability item.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  notes:  # $defs.notes_t
  - # <note-instance>:
    audience: String
    category: String.Enum
    group_ids: $defs.product_groups_t
    product_ids: $defs.products_t
    text: String
    title: String
  # ...
# ...
```

The following combinations of `category` and `title` have a special meaning and MUST be used as stated below:

`category`	`title`	content of `text`
`description`	CVE Description	Contains the official and unchanged CVE description for this specific vulnerability.
`description`	Preconditions	Contains a description of the preconditions that have to be fulfilled to be able to exploit the vulnerability, e.g. user account or physical access.
`summary`	Vulnerability Summary	Contains a summary of the vulnerability which is not the official CVE description.

If a note is specific to a product or product group it MUST be bound via the `group_ids` respectively `product_ids`.

3.2.4.12 Vulnerabilities Property - Product Status

Product status (`product_status`) of value type `object` with `1` or more properties contains different lists of `product_ids` which provide details on the status of the referenced product related to the current vulnerability. The nine defined properties are First affected (`first_affected`), First fixed (`first_fixed`), Fixed (`fixed`), Known affected (`known_affected`), Known not affected (`known_not_affected`), Last affected (`last_affected`), Recommended (`recommended`), Under investigation (`under_investigation`) and Unknown (`unknown`) are all of value type Products (`products_t`).

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  product_status:
    first_affected: $defs.products_t
    first_fixed: $defs.products_t
    fixed: $defs.products_t
```



```
    known_affected: $defs.products_t
    known_not_affected: $defs.products_t
    last_affected: $defs.products_t
    recommended: $defs.products_t
    under_investigation: $defs.products_t
    unknown: $defs.products_t
  # ...
```

First affected (`first_affected`) of value type Products (`products_t`) represents that these are the first versions of the releases known to be affected by the vulnerability.

First fixed (`first_fixed`) of value type Products (`products_t`) represents that these versions contain the first fix for the vulnerability but may not be the recommended fixed versions.

Fixed (`fixed`) of value type Products (`products_t`) represents that these versions contain a fix for the vulnerability but may not be the recommended fixed versions.

Known affected (`known_affected`) of value type Products (`products_t`) represents that these versions are known to be affected by the vulnerability. Actions are recommended to remediate or address this vulnerability.

> This could include for instance learning more about the vulnerability and context, and/or making a risk-based decision to patch or apply defense-in-depth measures. See `/vulnerabilities[]/remediations`, `/vulnerabilities[]/notes` and `/vulnerabilities[]/threats` for more details.

Known not affected (`known_not_affected`) of value type Products (`products_t`) represents that these versions are known not to be affected by the vulnerability. No remediation is required regarding this vulnerability.

> This could for instance be because the code referenced in the vulnerability is not present, not exposed, compensating controls exist, or other factors. See `/vulnerabilities[]/flags` and `/vulnerabilities[]/threats` in category `impact` for more details.

Last affected (`last_affected`) of value type Products (`products_t`) represents that these are the last versions in a release train known to be affected by the vulnerability. Subsequently released versions would contain a fix for the vulnerability.

Recommended (`recommended`) of value type Products (`products_t`) represents that these versions have a fix for the vulnerability and are the vendor-recommended versions for fixing the vulnerability.

Under investigation (`under_investigation`) of value type Products (`products_t`) represents that it is not known yet whether these versions are or are not affected by the vulnerability. However, it is still under investigation - the result will be provided in a later release of the document.

Unknown (`unknown`) of value type Products (`products_t`) represents that it is not known whether these versions are or are not affected by the vulnerability. There is also no investigation and therefore the status might never be determined.

The individual properties form the following product status groups:

- Affected:

  ```
  /vulnerabilities[]/product_status/first_affected[]
  /vulnerabilities[]/product_status/known_affected[]
  /vulnerabilities[]/product_status/last_affected[]
  ```

- Not affected:

  ```
  /vulnerabilities[]/product_status/known_not_affected[]
  ```

- Fixed:


```
/vulnerabilities[]/product_status/first_fixed[]
/vulnerabilities[]/product_status/fixed[]
```

- Under investigation:

```
/vulnerabilities[]/product_status/under_investigation[]
```

- Unknown:

```
/vulnerabilities[]/product_status/unknown[]
```

As the aforementioned product status groups contradict each other, the sets formed by the contradicting groups within one vulnerability item MUST be pairwise disjoint.

> Note: An issuer might recommend (`/vulnerabilities[]/product_status/recommended`) a product version from any group - also from the affected group, i.e. if it was discovered that fixed versions introduce a more severe vulnerability.

3.2.4.13 Vulnerabilities Property - References

Vulnerability references (`references`) of value type References Type (`references_t`) holds a list of references associated with this vulnerability item.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  references:  # $defs.references_t
  - # <reference-instance>:
    category: String.Enum
    summary: String
    url: String.URI
  # ...
# ...
```

3.2.4.14 Vulnerabilities Property - Remediations

List of remediations (`remediations`) of value type `array` with `1` or more Remediation items contains a list of remediations.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  remediations: Sequence
  # ...
```

Every Remediation item of value type `object` with the two mandatory properties Category (`category`) and Details (`details`) specifies details on how to handle (and presumably, fix) a vulnerability. At least one of the optional elements Group IDs (`group_ids`) and Product IDs (`product_ids`) MUST be present to state for which products or product groups this remediation is applicable.

In addition, any Remediation MAY expose the six optional properties Date (`date`), Entitlements (`entitlements`), Group IDs (`group_ids`), Product IDs (`product_ids`), Restart required (`restart_required`), and URL (`url`).


```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  remediations:
  - # <remediation-instance>:
    category: String.Enum
    date: String.DateTime
    details: String
    entitlements: Sequence
    group_ids: $defs.product_groups_t
    product_ids: $defs.products_t
    restart_required: Mapping
    url: String.URI
  # ...
```

3.2.4.14.1 Vulnerabilities Property - Remediations - Category

Category of the remediation (`category`) of value type `string` and `enum` specifies the category which this remediation belongs to. Valid values are:

```
fix_planned
mitigation
no_fix_planned
none_available
optional_patch
vendor_fix
workaround
```

The value `workaround` indicates that the remediation contains information about a configuration or specific deployment scenario that can be used to avoid exposure to the vulnerability. There MAY be none, one, or more workarounds available. This is typically the "first line of defense" against a new vulnerability before a mitigation or vendor fix has been issued or even discovered.

The value `mitigation` indicates that the remediation contains information about a configuration or deployment scenario that helps to reduce the risk of the vulnerability but that does not resolve the vulnerability on the affected product. Mitigations MAY include using devices or access controls external to the affected product. Mitigations MAY or MAY NOT be issued by the original author of the affected product, and they MAY or MAY NOT be officially sanctioned by the document producer.

The value `vendor_fix` indicates that the remediation contains information about an official fix that is issued by the original author of the affected product. Unless otherwise noted, it is assumed that this fix fully resolves the vulnerability.

The value `optional_patch` indicates that the remediation contains information about an patch that is issued by the original author of the affected product. Its application is not necessary, but might be desired by the user, e.g. to calm a security scanner by updating a dependency to a fixed version even though the dependency in the affected version was used in the product in a way that the product itself was not affected. Unless otherwise noted, it is assumed that this does not change the state regarding the vulnerability.

> This is sometimes also referred to as a "regulatory compliance patch".

The value `none_available` indicates that there is currently no fix or other remediation available. The text in field `details` SHOULD contain details about why there is no fix or other remediation.

The value `fix_planned` indicates that there is a fix for the vulnerability planned but not yet ready. An issuing party might choose to use this category to announce that a fix is currently developed. The text in field `details` SHOULD contain details including a date when a customer can expect the fix to be ready and distributed.

The value `no_fix_planned` indicates that there is no fix for the vulnerability and it is not planned to provide one at any time. This is often the case when a product has been orphaned, declared end-of-life, or otherwise deprecated. The text in field `details` SHOULD contain details about why there will be no fix issued.

Some category values contradict each other and thus are mutually exclusive per product. Therefore, such a combination MUST NOT be used in the list of remediations for the same product. This is independent from whether the product is referenced directly or indirectly through a product group. The following tables shows the allowed and prohibited combinations:

category value	workarour	mitigatic	vendor_f	optional_pa	none_availa	fix_plann	no_fix_planne
workaround	allowed	allowed	allowed	prohibited	prohibited	allowed	allowed
mitigation	allowed	allowed	allowed	prohibited	prohibited	allowed	allowed
vendor_fix	allowed	allowed	allowed	prohibited	prohibited	prohibited	prohibited
optional_patch	prohibited	prohibited	prohibited	allowed	prohibited	prohibited	prohibited
none_available	prohibited	prohibited	prohibited	prohibited	allowed	prohibited	prohibited
fix_planned	allowed	allowed	prohibited	prohibited	prohibited	allowed	prohibited
no_fix_planned	allowed	allowed	prohibited	prohibited	prohibited	prohibited	allowed

Table 1: Remediation Combinations

Some category values contradict certain product status groups. Therefore, such a combination MUST NOT exist in a vulnerability item for the same product. This is independent from whether the product is referenced directly or indirectly through a product group. The following tables shows the allowed, discouraged and prohibited combinations:

category value	Affected	Not Affected	Fixed	Under Investigation	Unknown	Recommended
workaround	allowed	prohibited	prohibited	discouraged	discouraged	allowed
mitigation	allowed	prohibited	prohibited	discouraged	discouraged	allowed
vendor_fix	allowed	prohibited	prohibited	discouraged	discouraged	allowed
optional_patch	prohibited	allowed	discouraged	allowed	allowed	allowed
none_available	allowed	prohibited	prohibited	allowed	allowed	allowed
fix_planned	allowed	discouraged	prohibited	discouraged	discouraged	allowed
no_fix_planned	allowed	discouraged	prohibited	allowed	allowed	allowed

Table 2: Product Status Remediation Category Combinations

The following preference for combinations of remediation categories and product status groups is RECOMMENDED:

1. `vendor_fix` and Recommended
2. `mitigation` and Recommended
3. `workaround` and Recommended
4. `optional_patch` and Recommended
5. `vendor_fix` and Affected
6. `mitigation` and Affected
7. `workaround` and Affected
8. `optional_patch` and Under Investigation
9. `optional_patch` and Unknown
10. `fix_planned` and Recommended
11. `fix_planned` and Affected
12. `optional_patch` and Not Affected
13. `none_available` and Recommended
14. `no_fix_planned` and Recommended
15. `none_available` and Affected
16. `none_available` and Under Investigation

17. `none_available` and Unknown
18. `no_fix_planned` and Affected
19. `no_fix_planned` and Under Investigation
20. `no_fix_planned` and Unknown

The remaining discouraged combinations are appended at the end of the list:

1. `optional_patch` and Fixed
2. `vendor_fix` and Under Investigation
3. `vendor_fix` and Unknown
4. `mitigation` and Under Investigation
5. `mitigation` and Unknown
6. `workaround` and Under Investigation
7. `workaround` and Unknown
8. `fix_planned` and Under Investigation
9. `fix_planned` and Unknown
10. `fix_planned` and Not Affected
11. `no_fix_planned` and Not Affected

CSAF Viewers MAY sort the remediation items accordingly.

3.2.4.14.2 Vulnerabilities Property - Remediations - Date

Date of the remediation (`date`) of value type `string` with format `date-time` contains the date from which the remediation is available.

3.2.4.14.3 Vulnerabilities Property - Remediations - Details

Details of the remediation (`details`) of value type `string` with `1` or more characters contains a thorough human-readable discussion of the remediation.

3.2.4.14.4 Vulnerabilities Property - Remediations - Entitlements

List of entitlements (`entitlements`) of value type `array` with `1` or more items of type Entitlement of the remediation as `string` with `1` or more characters contains a list of entitlements.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  remediations:
  - # <remediation-instance>:
    # ...
    entitlements: Sequence
    # ...
  # ...
```

Every Entitlement of the remediation contains any possible vendor-defined constraints for obtaining fixed software or hardware that fully resolves the vulnerability.

3.2.4.14.5 Vulnerabilities Property - Remediations - Group IDs

Group IDs (`group_ids`) are of value type Product Groups (`product_groups_t`) and contain a list of Product Groups the current remediation item applies to.

3.2.4.14.6 Vulnerabilities Property - Remediations - Product IDs

Product IDs (`product_ids`) are of value type Products (`products_t`) and contain a list of Products the current remediation item applies to.

3.2.4.14.7 Vulnerabilities Property - Remediations - Restart Required

Restart required by remediation (`restart_required`) of value type `object` with the one mandatory property Category (`category`) and the optional property Details (`details`) provides information on the category of restart required by this remediation to become effective.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  remediations:
  - # <remediation-instance>:
    # ...
    restart_required:
      category: String.Enum
      details: String
    # ...
  # ...
```

Category of restart (`category`) of value type `string` and `enum` specifies what category of restart is required by this remediation to become effective. Valid values are:

```
connected
dependencies
machine
none
parent
service
system
vulnerable_component
zone
```

The values MUST be used as follows:

- `none`: No restart required.
- `vulnerable_component`: Only the vulnerable component (as given by the elements of `product_ids` or `group_ids` in the current remediation item) needs to be restarted.
- `service`: The vulnerable component and the background service used by the vulnerable component need to be restarted.
- `parent`: The vulnerable component and its parent process need to be restarted. This could be the case if the parent process has no build-in way to restart the vulnerable component or process values / context is only given at the start of the parent process.
- `dependencies`: The vulnerable component and all components which require the vulnerable component to work need to be restarted. This could be the case e.g. for a core service of a software.
- `connected`: The vulnerable component and all components connected (via network or any type of inter-process communication) to the vulnerable component need to be restarted.
- `machine`: The machine on which the vulnerable component is installed on needs to be restarted. This is the value which SHOULD be used if an OS needs to be restarted. It is typically the case for OS upgrades.

- `zone`: The security zone in which the machine resides on which the vulnerable component is installed needs to be restarted. This value might be useful for a remediation if no patch is available. If the malware can be wiped out by restarting the infected machines but the infection spreads fast the controlled shutdown of all machines at the same time and restart afterwards can leave one with a clean system.
- `system`: The whole system which the machine resides on which the vulnerable component is installed needs to be restarted. This MAY include multiple security zones. This could be the case for a major system upgrade in an ICS system or a protocol change.

Additional restart information (`details`) of value type `string` with `1` or more characters provides additional information for the restart. This can include details on procedures, scope or impact.

3.2.4.14.8 Vulnerabilities Property - Remediations - URL

URL (`url`) of value type `string` with format `uri` contains the URL where to obtain the remediation.

3.2.4.15 Vulnerabilities Property - Threats

List of threats (`threats`) of value type `array` with `1` or more items of value type `object` contains information about a vulnerability that can change with time.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  threats: Sequence
  # ...
# ...
```

Every Threat item of value type `object` with the two mandatory properties Category (`category`) and Details (`details`) contains the vulnerability kinetic information. This information can change as the vulnerability ages and new information becomes available. In addition, any Threat item MAY expose the three optional properties Date (`date`), Group IDs (`group_ids`), and Product IDs (`product_ids`).

```
yaml <csaf-instance>:   # ...   vulnerabilities:   - # <vulnerability-instance>:     #
...     threats:       category: String.Enum       date: String.DateTime       details:
String       group_ids: $defs.product_groups_t       product_ids: $defs.products_t
...   # ...
```

Category of the threat (`category`) of value type `string` and `enum` categorizes the threat according to the rules of the specification. Valid values are:

```
exploit_status
impact
target_set
```

The value `exploit_status` indicates that the `details` field contains a description of the degree to which an exploit for the vulnerability is known. This knowledge can range from information privately held among a very small group to an issue that has been described to the public at a major conference or is being widely exploited globally. For consistency and simplicity, this section can be a mirror image of the CVSS `exploitMaturity` (v4.0), respectively `exploitCodeMaturity` (v3.1 and v3.0) or `exploitability` (v2.0) metric. However, it can also contain a more contextual status, such as “Weaponized” or “Functioning Code”.

The value `impact` indicates that the `details` field contains an assessment of the impact on the user or the target set if the vulnerability is successfully exploited or a description why it cannot be exploited. If applicable, for consistency and simplicity, this section can be a textual summary of the three CVSS impact metrics. These metrics measure how a

vulnerability detracts from the three core security properties of an information system: Confidentiality, Integrity, and Availability.

The value `target_set` indicates that the `details` field contains a description of the currently known victim population in whatever terms are appropriate. Such terms MAY include: operating system platform, types of products, user segments, and geographic distribution.

Date of the threat (`date`) of value type `string` with format `date-time` contains the date when the assessment was done or the threat appeared.

Details of the threat (`details`) of value type `string` with `1` or more characters represents a thorough human-readable discussion of the threat.

Group IDs (`group_ids`) are of value type Product Groups (`product_groups_t`) and contain a list of Product Groups the current threat item applies to.

Product IDs (`product_ids`) are of value type Products (`products_t`) and contain a list of Products the current threat item applies to.

3.2.4.16 Vulnerabilities Property - Title

Title (`title`) has value type `string` with `1` or more characters and gives the document producer the ability to apply a canonical name or title to the vulnerability.

3.2.4.17 Vulnerabilities Property - Extensions

Vulnerability-level Extensions (`x_extensions`) of value type Extensions Type (`extensions_t`) contains a list of extensions valid at the vulnerability item level of the CSAF document and associated with this vulnerability element.

```
# ...
vulnerabilities:
- # <vulnerability-instance>:
  # ...
  x_extensions:  # $defs.extensions_t
  - # <x_extension-instance>:
    $schema: String
    category: String.Enum
    content: Mapping
    critical: Boolean
    # ...
```

3.2.5 Extensions Property

Root-level Extensions (`x_extensions`) of value type Extensions Type (`extensions_t`) contains a list of extensions valid at the root-level of the CSAF document and associated with this CSAF document.

```
# ...
x_extensions:  # $defs.extensions_t
- # <x_extension-instance>:
  $schema: String
  category: String.Enum
  content: Mapping
  critical: Boolean
  # ...
```


4 Profiles

CSAF documents do not have many required fields as they can be used for different purposes. To ensure a common understanding of which fields are required in a given use case the standard defines profiles. Each subsection describes such a profile by describing necessary content for that specific use case and providing insights into its purpose. The value of `/document/category` is used to identify a CSAF document's profile. The following rules apply:

1. Each CSAF document MUST conform the **CSAF Base** profile.
2. Each profile extends the base profile "CSAF Base" - directly or indirect through another profile from the standard - by making additional fields from the standard mandatory. A profile can always add, but never subtract nor overwrite requirements defined in the profile it extends.
3. Any optional field from the standard can also be added to a CSAF document which conforms with a profile without breaking conformance with the profile. One and only exempt is when the profile requires not to have a certain set of fields.
4. Values of `/document/category` starting with `csaf_` are reserved for existing, past, upcoming and future profiles defined in the CSAF standard.
5. Values of `/document/category` starting with `csaf_deprecated_` are used for official profiles that are marked deprecated. Those profiles are mostly there to allow backwards compatibility, e.g. with older CSAF versions. Therefore, they SHOULD NOT be used for newly created CSAF documents.
6. Values of `/document/category` that do not match any of the values defined in section 4 of this standard SHALL be validated against the "CSAF Base" profile.
7. Local or private profiles MAY exist and tools MAY choose to support them.
8. If an official profile and a private profile exists, tools MUST validate against the official one from the standard.

4.1 Profile 1: CSAF Base

This profile defines the default required fields for any CSAF document. Therefore, it is a "catch all" for CSAF documents that do not satisfy any other profile. Furthermore, it is the foundation all other profiles are build on.

A CSAF document SHALL fulfill the following requirements to satisfy the profile "CSAF Base":

- The following elements MUST exist and be valid:
  - `/$schema`
  - `/document/category`
  - `/document/csaf_version`
  - `/document/distribution/tlp/label`
  - `/document/publisher/category`
  - `/document/publisher/name`
  - `/document/publisher/namespace`
  - `/document/title`
  - `/document/tracking/current_release_date`
  - `/document/tracking/id`
  - `/document/tracking/initial_release_date`
  - `/document/tracking/revision_history[]/date`
  - `/document/tracking/revision_history[]/number`
  - `/document/tracking/revision_history[]/summary`
  - `/document/tracking/status`
  - `/document/tracking/version`
- The value of `/document/category` SHALL NOT be equal to or a close match for any value that is intended to only be used by another profile nor to the (case insensitive) name of any other profile from the standard. Such case insensitive matching does not take occurrences of dash, hyphen, minus, white space, and underscore characters into account. To explicitly select the use of this profile the value `csaf_base` SHOULD be used.

> Neither `CSAF Security Advisory` nor `csaf security advisory` are valid values for `/document/category`.

An issuing party might choose to set `/document/publisher/name` in front of a value that is intended to only be used by another profile to state that the CSAF document does not use the profile associated with this value. In this case, the (case insensitive) string "CSAF" MUST be removed from the value. This SHOULD be done if the issuing party is unable or unwilling to use the value `csaf_base`, e.g. due to legal or cooperate identity reasons.

> Both values `Example Company Security Advisory` and `Example Company security_advisory` in `/document/category` use the profile "CSAF Base". This is important to prepare forward compatibility as later versions of CSAF might add new profiles. Therefore, the values which can be used for the profile "CSAF Base" might change.

4.2 Profile 2: Security Incident Response

This profile SHOULD be used to provide a response to a security breach or incident. This MAY also be used to convey information about an incident that is unrelated to the issuing party's own products or infrastructure.

> Example Company might use a CSAF document satisfying this profile to respond to a security incident at ACME Inc. and the implications on its own products and infrastructure.

A CSAF document SHALL fulfill the following requirements to satisfy the profile "Security Incident Response":

- The following elements MUST exist and be valid:
  - all elements required by the profile "CSAF Base".
  - `/document/notes` with at least one item which has a `category` of `description`, `details`, `general` or `summary`

    > Reasoning: Without at least one note item which contains information about response to the event referred to this doesn't provide any useful information.

  - `/document/references` with at least one item which has a `category` of `external`

    > The intended use for this field is to refer to one or more documents or websites which provides more details about the incident.

- The value of `/document/category` SHALL be `csaf_security_incident_response`.

4.3 Profile 3: Informational Advisory

This profile SHOULD be used to provide information which are **not related to a vulnerability** but e.g. a misconfiguration.

A CSAF document SHALL fulfill the following requirements to satisfy the profile "Informational Advisory":

- The following elements MUST exist and be valid:
  - all elements required by the profile "CSAF Base".
  - `/document/notes` with at least one item which has a `category` of `description`, `details`, `general` or `summary`

    > Reasoning: Without at least one note item which contains information about the "issue" which is the topic of the advisory it is useless.

  - `/document/references` with at least one item which has a `category` of `external`

> The intended use for this field is to refer to one or more documents or websites which provide more details about the issue or its remediation (if possible). This could be a hardening guide, a manual, best practices or any other helpful information.

- The value of `/document/category` SHALL be `csaf_informational_advisory`.
- The element `/vulnerabilities` SHALL NOT exist. If there is any information that would reside in the element `/vulnerabilities` the CSAF document SHOULD use another profile, e.g. “Security Advisory”.

If the element `/product_tree` exists, a user MUST assume that all products mentioned are affected.

4.4 Profile 4: Security Advisory

This profile SHOULD be used to provide information which is related to vulnerabilities and corresponding remediations.

A CSAF document SHALL fulfill the following requirements to satisfy the profile “Security Advisory”:

- The following elements MUST exist and be valid:
  - all elements required by the profile “CSAF Base”.
  - `/product_tree` which lists all products referenced later on in the CSAF document regardless of their state.
  - `/vulnerabilities` which lists all vulnerabilities.
  - `/vulnerabilities[]/notes`

    > Provides details about the vulnerability.

  - `/vulnerabilities[]/product_status`

    > Lists each product’s status in regard to the vulnerability.

  - `/vulnerabilities[]/product_status/known_affected`

    > Lists affected products in regard to the vulnerability.

  - For each product given in `/vulnerabilities[]/product_status/fixed`, the corresponding affected version SHALL be given.

    > Corresponding versions are usually in the same `branches` element.

- The value of `/document/category` SHALL be `csaf_security_advisory`.
- The following elements SHOULD exist:
  - `/vulnerabilities[]/product_status/fixed`

    > Lists fixed products in regard to the vulnerability.

  - `/vulnerabilities[]/remediations`

    > Lists for each affected product in regard to the vulnerability appropriate remediations.

4.5 Profile 5: VEX

This profile SHOULD be used to provide information of the “Vulnerability Exploitability eXchange”. The main purpose of the VEX format is to state that and why a certain product is, or is not, affected by a vulnerability. See [VEX] for details.

A CSAF document SHALL fulfill the following requirements to satisfy the profile “VEX”:

- The following elements MUST exist and be valid:

  - all elements required by the profile “CSAF Base”.
  - `/product_tree` which lists all products referenced later on in the CSAF document regardless of their state.
  - `/vulnerabilities` which lists all vulnerabilities.
  - at least one of
    * `/vulnerabilities[]/product_status/fixed`
    * `/vulnerabilities[]/product_status/known_affected`
    * `/vulnerabilities[]/product_status/known_not_affected`
    * `/vulnerabilities[]/product_status/under_investigation`
  - at least one of
    * `/vulnerabilities[]/cve`
    * `/vulnerabilities[]/ids`
  - `/vulnerabilities[]/notes`

    > Provides details about the vulnerability.

- For each item in
  - `/vulnerabilities[]/product_status/known_not_affected` an impact statement SHALL exist as machine readable flag in `/vulnerabilities[]/flags` or as human readable justification in `/vulnerabilities[]/threats`. For the latter one, the `category` value for such a statement MUST be `impact` and the `details` field SHALL contain a description why the vulnerability cannot be exploited.
  - `/vulnerabilities[]/product_status/known_affected` additional product specific information SHALL be provided in `/vulnerabilities[]/remediations` as an action statement. Optional, additional information MAY also be provide through `/vulnerabilities[]/notes` and `/vulnerabilities[]/threats`.

    > The use of the categories `no_fix_planned` and `none_available` for an action statement is permitted.

  > Even though Product status lists Product IDs, Product Group IDs can be used in the `remediations` and `threats` object. However, it MUST be ensured that for each Product ID the required information according to its product status as stated in the two points above is available. This implies that all products with the status `known_not_affected` MUST have an impact statement and all products with the status `known_affected` MUST have additional product specific information regardless of whether that is referenced through the Product ID or a Product Group ID.

- The value of `/document/category` SHALL be `csaf_vex`.

4.6 Profile 6: Deprecated Security Advisory

This profile MAY be used to provide information which is related to vulnerabilities and corresponding remediations, e.g. when converting CSAF documents from older CSAF versions or a human-readable format. It SHOULD NOT be used for newly created documents. The profile “Security Advisory” from section 4.4 SHOULD be used instead.

> The definition of the profile “Deprecated Security Advisory” in CSAF 2.1 matches the definition of profile “Security Advisory” in CSAF 2.0.

A CSAF document SHALL fulfill the following requirements to satisfy the profile “Deprecated Security Advisory”:

- The following elements MUST exist and be valid:
  - all elements required by the profile “CSAF Base”.
  - `/product_tree` which lists all products referenced later on in the CSAF document regardless of their state.

  - `/vulnerabilities` which lists all vulnerabilities.
  - `/vulnerabilities[]/notes`

    > Provides details about the vulnerability.

  - `/vulnerabilities[]/product_status`

    > Lists each product's status in regard to the vulnerability.

- The value of `/document/category` SHALL be `csaf_deprecated_security_advisory`.

4.7 Profile 7: Withdrawn

This profile MUST be used for any CSAF document that is withdrawn. It MUST NOT be used for any superseded document.

A CSAF document SHALL fulfill the following requirements to satisfy the profile "Withdrawn":

- The following elements MUST exist and be valid:
  - all elements required by the profile "CSAF Base".
  - `/document[]/notes` with exactly one item using the `category` `description` describing the original content and the reasons for the withdrawal

    > Other items, such as a legal disclaimer, may exist alongside the required one.

    The `title` MUST be `Reasoning for Withdrawal` for English or an unspecified document language. For any other language, it SHOULD be the language specific translation of that term.
  - `/document/tracking/revision_history` with at least 2 entries. Any previous items MUST NOT be removed.

    > A CSAF document cannot be withdrawn during the initial release to its specified target group. In such case, the CSAF document should not be released at all. If it was shared previously in draft status, then the `/document/tracking/status` is kept in `draft`.

- The value of `/document/category` SHALL be `csaf_withdrawn`.
- The elements `/product_tree` and `/vulnerabilities` SHALL NOT exist.

The CSAF document MAY link to additional information through `/document/references`.

4.8 Profile 8: Superseded

This profile MUST be used for any CSAF document that is superseded. It MUST NOT be used for any withdrawn document.

A CSAF document SHALL fulfill the following requirements to satisfy the profile "Superseded":

- The following elements MUST exist and be valid:
  - all elements required by the profile "CSAF Base".
  - `/document[]/notes` with exactly one item using the `category` `description`

    > Other items, such as a legal disclaimer, may exist alongside the required one.

    The `title` MUST be `Reasoning for Supersession` for English or an unspecified document language. For any other language, it SHOULD be the language specific translation of that term.
  - `/document/tracking/revision_history` with at least 2 entries. Any previous items MUST NOT be removed.

> A CSAF document cannot be superseded during the initial release to its specified target group. In such case, the CSAF document should not be released at all. If it was shared previously in draft status, then the `/document/tracking/status` is kept in `draft`.

  - `/document/references` containing at least one item with `category` `external` The `summary` MUST start with `Superseding Document` for English or an unspecified document language. For any other language, it SHOULD be the language specific translation of that term.

- The value of `/document/category` SHALL be `csaf_superseded`.
- The elements `/product_tree` and `/vulnerabilities` SHALL NOT exist.

5 Additional Conventions

This section provides additional rules for handling CSAF documents.

5.1 Filename

The following rules MUST be applied to determine the filename for the CSAF document:

1. The value `/document/tracking/id` is converted into lowercase.
2. Any character sequence which is not part of one of the following groups MUST be replaced by a single underscore (_):
   - lower-case ASCII letters (0x61 - 0x7A)
   - digits (0x30 - 0x39)
   - special characters: + (0x2B), − (0x2D)

   > The regex `[^+\-a-z0-9]+` can be used to find a character sequence which has to be replaced by an underscore. However, it SHALL NOT be applied before completing the first step.
   > Even though the underscore _ (0x5F) is a valid character in the filename it is replaced to avoid situations where the conversion rule might lead to multiple consecutive underscores. As a result, a `/document/tracking/id` with the value `2022_#01-A` is converted into `2022_01-a` instead of `2022__01-a`.
3. The file extension `.json` MUST be appended.

Examples 1:

```
cisco-sa-20190513-secureboot.json
example_company_-_2019-yh3234.json
rhba-2019_0024.json
```

> It is currently considered best practice to indicate that a CSAF document is invalid by inserting `_invalid` into the filename in front of the file extension.

Examples 2:

```
cisco-sa-20190513-secureboot_invalid.json
example_company_-_2019-yh3234_invalid.json
rhba-2019_0024_invalid.json
```

5.2 Separation in Data Stream

If multiple CSAF documents are transported via a data stream in a sequence without requests inbetween, they MUST be separated by the Record Separator in accordance with [RFC7464].

5.3 Sorting

The keys within a CSAF document SHOULD be sorted alphabetically.

5.4 Usage of Markdown

The use of GitHub-flavoured Markdown is permitted in the following fields:

```
/document/acknowledgments[]/summary
/document/distribution/text
/document/notes[]/text
/document/publisher/issuing_authority
/document/references[]/summary
/document/tracking/revision_history[]/summary
/product_tree/product_groups[]/summary
/vulnerabilities[]/acknowledgments[]/summary
/vulnerabilities[]/involvements[]/summary
/vulnerabilities[]/notes[]/text
/vulnerabilities[]/references[]/summary
/vulnerabilities[]/remediations[]/details
/vulnerabilities[]/remediations[]/entitlements[]
/vulnerabilities[]/remediations[]/restart_required/details
/vulnerabilities[]/threats[]/details
```

Other fields MUST NOT contain Markdown.

5.5 Branch Recursion

The `/product_tree` uses a nested structure for `branches`. Along a single path to a leaf, the recursion of `branches` is limited to 30 repetitions. Therefore, the longest path to a leaf is:

```
/product_tree/branches[]/branches[]/branches[]/branches[]/branches[]/branches[]/branche
```

5.6 Hardware and Software within the Product Tree

If a product consists of hardware and software, the hardware part MUST be presented as one product in the product tree and the software part as another one. To form the overall product, both parts MUST be combined through a product path.

Example 1:

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version",
              "name": "1.0",
              "product": {
                "name": "Example Company Controller A 1.0",
                "product_id": "CSAFPID-908070601",
                "product_identification_helper": {
                  "serial_numbers": [
                    "143-D-354"
                  ]
```



```
          }
        }
      }
    ],
    "category": "product_name",
    "name": "Controller A"
  },
  {
    "branches": [
      {
        "category": "product_version",
        "name": "4.1",
        "product": {
          "name": "Example Company Controller A Firmware 4.1",
          "product_id": "CSAFPID-908070602",
          "product_identification_helper": {
            "hashes": [
              {
                "file_hashes": [
                  {
                    "algorithm": "sha256",
                    "value": "3fb9d502d096b1dfbcdfe60eed80ddecd98c8771bf21a82bb
                  }
                ],
                "filename": "a_4-1.bin"
              }
            ]
          }
        }
      },
      {
        "category": "product_version",
        "name": "4.2",
        "product": {
          "name": "Example Company Controller A Firmware 4.2",
          "product_id": "CSAFPID-908070603",
          "product_identification_helper": {
            "hashes": [
              {
                "file_hashes": [
                  {
                    "algorithm": "sha256",
                    "value": "0a853ce2337f0608489ac596a308dc5b7b19d35a52b10bf31
                  }
                ],
                "filename": "a_4-2.bin"
              }
            ]
          }
        }
      }
    ],
    "category": "product_name",
```



```
          "name": "Controller A Firmware"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ],
  "product_paths": [
    {
      "beginning_product_reference": "CSAFPID-908070602",
      "full_product_name": {
        "name": "Example Company Controller A Firmware 4.1 installed on Example Compa
        "product_id": "CSAFPID-908070604"
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070601"
        }
      ]
    },
    {
      "beginning_product_reference": "CSAFPID-908070603",
      "full_product_name": {
        "name": "Example Company Controller A Firmware 4.2 installed on Example Compa
        "product_id": "CSAFPID-908070605"
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070601"
        }
      ]
    }
  ]
}
```

This requirement is important to allow for correct matching. The serial number `143-D-354` identifies the `Example Company Controller A 1.0` which is in this example the hardware in its version 1.0. The hash `3fb9d502d096b1dfbcdfe60eed80ddecd98c8771bf21a82bbe1752735c4dc9e2` identifies the software in the version 4.1; the hash `0a853ce2337f0608489ac596a308dc5b7b19d35a52b10bf31261586ac368b` identifies the software in the version 4.2. The product paths combine the software and hardware part and form new products. These are used e.g. to assign the product status in the vulnerability section.
A matching tool searches in a first step for the product identification helper, e.g the serial number in an asset database to identify the asset that has this specific hardware and matches the software separately in a second step.
Representing the software version as a child element under elements representing hardware unsettles the consumer whether the version applies to the software or hardware. Also, this would violate the rule regarding the full identification of a product by the `product_identification_helper` from section 3.1.4.3.
Based on the CVE statistics up to and including the year 2024, in the majority of cases the vulnerabilities reside in software or are remediated via software. Having multiple products with the same `product_identification_helper` in different `product_status` for the same vulnerability would make it undecidable for machines what the `product_status` actually is.

6 Tests

The first three subsections list a number of tests which all will have a short description and an excerpt of an example which fails the test. The fourth subsection groups tests into preset.

6.1 Mandatory Tests

Mandatory tests MUST NOT fail at a valid CSAF document. A program MUST handle a test failure as an error.

6.1.1 Missing Definition of Product ID

For each element of type `/$defs/product_id_t` which is not inside a Full Product Name (type: `full_product_name_t`) and therefore reference an element within the `product_tree` it MUST be tested that the Full Product Name element with the matching `product_id` exists. The same applies for all items of elements of type `/$defs/products_t`.

The relevant paths for this test are:

```
/document/notes[]/product_ids[]
/product_tree/product_groups[]/product_ids[]
/product_tree/product_paths[]/beginning_product_reference
/product_tree/product_paths[]/subpaths[]/next_product_reference
/vulnerabilities[]/first_known_exploitation_dates[]/product_ids[]
/vulnerabilities[]/flags[]/product_ids[]
/vulnerabilities[]/involvements[]/product_ids[]
/vulnerabilities[]/metrics[]/products[]
/vulnerabilities[]/notes[]/product_ids[]
/vulnerabilities[]/product_status/first_affected[]
/vulnerabilities[]/product_status/first_fixed[]
/vulnerabilities[]/product_status/fixed[]
/vulnerabilities[]/product_status/known_affected[]
/vulnerabilities[]/product_status/known_not_affected[]
/vulnerabilities[]/product_status/last_affected[]
/vulnerabilities[]/product_status/recommended[]
/vulnerabilities[]/product_status/under_investigation[]
/vulnerabilities[]/product_status/unknown[]
/vulnerabilities[]/remediations[]/product_ids[]
/vulnerabilities[]/threats[]/product_ids[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "product_groups": [
    {
      "group_id": "CSAFGID-1020300",
      "product_ids": [
        "CSAFPID-9080700",
        "CSAFPID-9080701"
      ]
    }
  ]
}
```


Neither `CSAFPID-9080700` nor `CSAFPID-9080701` were defined in the `product_tree`.

6.1.2 Multiple Definition of Product ID

For each Product ID (type `/$defs/product_id_t`) in Full Product Name elements (type: `/$defs/full_product_name_t`) it MUST be tested that the `product_id` was not already defined within the same document.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_id
/product_tree/full_product_names[]/product_id
/product_tree/product_paths[]/full_product_name/product_id
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    },
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product B"
    }
  ]
}
```

`CSAFPID-9080700` was defined twice.

6.1.3 Circular Definition of Product ID

For each new defined Product ID (type `/$defs/product_id_t`) in items of product paths (`/product_tree/product_pat`
it MUST be tested that the `product_id` does not end up in a circle.

The relevant path for this test is:

```
/product_tree/product_paths[]/full_product_name/product_id
```

As this can be quite complex a program for large CSAF documents, a program could check first whether a Product ID defined in a product path item is used as `beginning_product_reference` or `next_product_reference`. Only for those which fulfill this condition it is necessary to run the full check following the references.

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700"
```



```
      }
    ],
    "product_paths": [
      {
        "beginning_product_reference": "CSAFPID-9080700",
        "full_product_name": {
          "name": "Product B",
          "product_id": "CSAFPID-9080701"
        },
        "subpaths": [
          {
            "category": "installed_on",
            "next_product_reference": "CSAFPID-9080701"
          }
        ]
      }
    ]
  }
```

`CSAFPID-9080701` refers to itself - this is a circular definition.

6.1.4 Missing Definition of Product Group ID

For each element of type `/$defs/product_group_id_t` which is not inside a Product Group (`/product_tree/product` and therefore reference an element within the `product_tree` it MUST be tested that the Product Group element with the matching `group_id` exists. The same applies for all items of elements of type `/$defs/product_groups_t`.

The relevant paths for this test are:

```
/document/notes[]/group_ids[]
/vulnerabilities[]/first_known_exploitation_dates[]/group_ids[]
/vulnerabilities[]/flags[]/group_ids[]
/vulnerabilities[]/involvements[]/group_ids[]
/vulnerabilities[]/notes[]/group_ids[]
/vulnerabilities[]/remediations[]/group_ids[]
/vulnerabilities[]/threats[]/group_ids[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700"
    }
  ]
},
"vulnerabilities": [
  {
    "threats": [
      {
        "category": "exploit_status",
```



```
        "details": "Reliable exploits integrated in Metasploit.",
        "group_ids": [
          "CSAFGID-1020301"
        ]
      }
    ]
  }
]
```

`CSAFGID-1020301` was not defined in the Product Tree.

6.1.5 Multiple Definition of Product Group ID

For each Product Group ID (type `/$defs/product_group_id_t`) Product Group elements (`/product_tree/product_groups[]`) it MUST be tested that the `group_id` was not already defined within the same document.

The relevant path for this test is:

```
/product_tree/product_groups[]/group_id
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    },
    {
      "product_id": "CSAFPID-9080701",
      "name": "Product B"
    },
    {
      "product_id": "CSAFPID-9080702",
      "name": "Product C"
    }
  ],
  "product_groups": [
    {
      "group_id": "CSAFGID-1020300",
      "product_ids": [
        "CSAFPID-9080700",
        "CSAFPID-9080701"
      ]
    },
    {
      "group_id": "CSAFGID-1020300",
      "product_ids": [
        "CSAFPID-9080700",
        "CSAFPID-9080702"
      ]
```



```
      }
    ]
  }
```

`CSAFGID-1020300` was defined twice.

6.1.6 Contradicting Product Status

For each item in `/vulnerabilities` it MUST be tested that the same Product ID is not a member of contradicting product status groups (see section 3.2.4.12). The sets formed by the contradicting groups within one vulnerability item MUST be pairwise disjoint.

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "product_status": {
      "known_affected": [
        "CSAFPID-9080700"
      ],
      "known_not_affected": [
        "CSAFPID-9080700"
      ]
    }
  }
]
```

`CSAFPID-9080700` is a member of the two contradicting groups “Affected” and “Not affected”.

6.1.7 Multiple Scores with Same Version per Product

For each item in `/vulnerabilities` it MUST be tested that the same Product ID is not a member of more than one CVSS vector with the same version and same source.

Different sources might assign different scores for the same product.

The test also applies for versions not included in CSAF.

The relevant path for this test is:


```
/vulnerabilities[]/metrics[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "cvss_v3": {
            "version": "3.1",
            "vectorString": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H",
            "baseScore": 10,
            "baseSeverity": "CRITICAL"
          }
        },
        "products": [
          "CSAFPID-9080700"
        ]
      },
      {
        "content": {
          "cvss_v3": {
            "version": "3.1",
            "vectorString": "CVSS:3.1/AV:L/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:H",
            "baseScore": 6.5,
            "baseSeverity": "MEDIUM"
          }
        },
        "products": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

Two CVSS v3.1 scores are given for `CSAFPID-9080700` by the document author.

6.1.8 Invalid CVSS

It MUST be tested that the given CVSS object is valid according to the referenced schema.

The relevant paths for this test are:


```
/vulnerabilities[]/metrics[]/content/cvss_v2
/vulnerabilities[]/metrics[]/content/cvss_v3
/vulnerabilities[]/metrics[]/content/cvss_v4
```

Example 1 (which fails the test):

```
"cvss_v3": {
  "version": "3.1",
  "vectorString": "CVSS:3.1/AV:L/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:H",
  "baseScore": 6.5
}
```

The required element `baseSeverity` is missing.

A tool MAY add one or more of the missing properties `version`, `baseScore` and `baseSeverity` based on the values given in `vectorString` as a quick fix.

6.1.9 Invalid CVSS Computation

It MUST be tested that the given CVSS object has the values computed correctly according to the definition.

The `vectorString` SHOULD take precedence.

The relevant paths for this test are:

```
/vulnerabilities[]/metrics[]/content/cvss_v2/baseScore
/vulnerabilities[]/metrics[]/content/cvss_v2/temporalScore
/vulnerabilities[]/metrics[]/content/cvss_v2/environmentalScore
/vulnerabilities[]/metrics[]/content/cvss_v3/baseScore
/vulnerabilities[]/metrics[]/content/cvss_v3/baseSeverity
/vulnerabilities[]/metrics[]/content/cvss_v3/temporalScore
/vulnerabilities[]/metrics[]/content/cvss_v3/temporalSeverity
/vulnerabilities[]/metrics[]/content/cvss_v3/environmentalScore
/vulnerabilities[]/metrics[]/content/cvss_v3/environmentalSeverity
/vulnerabilities[]/metrics[]/content/cvss_v4/baseScore
/vulnerabilities[]/metrics[]/content/cvss_v4/baseSeverity
```

Note: CVSS v4 does not define `threatScore`, `threatSeverity`, `environmentalScore` and `environmentalSeverity`. The existence of these JSON keys in an older version of the schema was fixed with version `4.0.1`.

Example 1 (which fails the test):

```
"cvss_v3": {
  "version": "3.1",
  "vectorString": "CVSS:3.1/AV:L/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:H",
  "baseScore": 10.0,
  "baseSeverity": "LOW"
}
```

Neither `baseScore` nor `baseSeverity` has the correct value according to the specification.

A tool MAY set the correct values as computed according to the specification as a quick fix.

6.1.10 Inconsistent CVSS

It MUST be tested that the given CVSS properties do not contradict the CVSS vector.

The relevant paths for this test are:

```
/vulnerabilities[]/metrics[]/content/cvss_v2
/vulnerabilities[]/metrics[]/content/cvss_v3
/vulnerabilities[]/metrics[]/content/cvss_v4
```

Example 1 (which fails the test):

```
"cvss_v3": {
  "version": "3.1",
  "vectorString": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H",
  "baseScore": 9.8,
  "baseSeverity": "CRITICAL",
  "attackVector": "LOCAL",
  "attackComplexity": "LOW",
  "privilegesRequired": "NONE",
  "userInteraction": "NONE",
  "scope": "CHANGED",
  "confidentialityImpact": "HIGH",
  "integrityImpact": "HIGH",
  "availabilityImpact": "LOW"
}
```

The values in the CVSS vector differ from values of the properties `attackVector`, `scope` and `availabilityImpact`.

A tool MAY overwrite contradicting values according to the `vectorString` as a quick fix.

6.1.11 CWE

For each CWE it MUST be tested that the given CWE exists and is valid in the `version` provided. Any `id` that refers to a CWE Category or View MUST fail the test.

A list of all CWE release archives is available at [[CWE-A]](#CWE-A)].

The relevant path for this test is:

```
/vulnerabilities[]/cwes[]
```

Example 1 (which fails the test):


```
"cwes": [
  {
    "id": "CWE-79",
    "name": "Improper Input Validation",
    "version": "4.13"
  }
]
```

The `CWE-79` exists. However, its name in version `4.13` is `Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')`.

6.1.12 Language

For each element of type `/$defs/lang_t` it MUST be tested that the language code is valid and exists.

The relevant paths for this test are:

```
/document/lang
/document/source_lang
```

Example 1 (which fails the test):

```
"lang": "EZ"
```

`EZ` is not a valid language. It is the subtag for the region "Eurozone".

For any deprecated subtag, a tool MAY replace it with its preferred value as a quick fix.

6.1.13 PURL

It MUST be tested that all given PURLs are valid.

It is not sufficient to just test against the `pattern` provided in section 3.1.4.3.4. The PURL must be validated against the requirements in the [ECMA-427] specification and the additional constraints given in section 3.1.4.3.4.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/purls[]
/product_tree/full_product_names[]/product_identification_helper/purls[]
/product_tree/product_paths[]/full_product_name/product_identification_helper/purls[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700",
      "product_identification_helper": {
        "purls": [
```



```
            "pkg:maven/@1.3.4"
          ]
        }
      }
    ]
  }
```

Any valid PURL has a name component.

6.1.14 Sorted Revision History

It MUST be tested that the value of `number` of items of the revision history are sorted ascending when the items are sorted ascending by `date` and as a second level criteria `number`. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The relevant path for this test is:

```
/document/tracking/revision_history
```

Example 1 (which fails the test):

```
"revision_history": [
  {
    "date": "2024-01-22T10:00:00.000Z",
    "number": "2",
    "summary": "Second version."
  },
  {
    "date": "2024-01-23T10:00:00.000Z",
    "number": "1",
    "summary": "Initial version."
  }
]
```

The first item has a higher version number than the second.

6.1.15 Translator

It MUST be tested that `/document/source_lang` is present and set if the value `translator` is used for `/document/publisher/category`. A CSAF Validator SHALL differentiate in the error message between the key being present but having no or an empty value and not being present at all.

The relevant path for this test is:

```
/document/source_lang
```

Example 1 (which fails the test):


```
"document": {
  // ...
  "publisher": {
    "category": "translator",
    "name": "CSAF TC Translator",
    "namespace": "https://csaf.io/translator"
  },
  "title": "Mandatory test: Translator (failing example 1)",
  // ...
}
```

The required element `source_lang` is missing.

A tool MAY add the key as a quick fix. In such case, the value still needs to be set - either manually or by other means from the tool, e.g. through a given configuration.

6.1.16 Latest Document Version

It MUST be tested that document version has the same value as the `number` in the last item of the revision history when it is sorted ascending by `date` and as a second level criteria `number`. As the timestamps might use different timezones, the sorting MUST take timezones into account. Build metadata is ignored in the comparison. Any pre-release part is also ignored if the document status is `draft`.

The relevant path for this test is:

```
/document/tracking/version
```

Example 1 (which fails the test):

```
"tracking": {
  // ...
  "revision_history": [
    {
      "date": "2024-01-21T09:00:00.000Z",
      "number": "1",
      "summary": "Initial version."
    },
    {
      "date": "2024-01-21T10:00:00.000Z",
      "number": "2",
      "summary": "Second version."
    }
  ],
  // ...
  "version": "1"
}
```

The value of `number` of the last item after sorting is `2`. However, the document version is `1`.

A tool MAY set the document version to the value of `number` of the last item in the revision history after sorting as a quick fix.

6.1.17 Document Status Draft

It MUST be tested that document status is `draft` if the document version is `0` or `0.y.z` or contains the pre-release part.

The relevant path for this test is:

```
/document/tracking/status
```

Example 1 (which fails the test):

```
"tracking": {
  // ...
  "status": "final",
  "version": "0.9.5"
}
```

The `/document/tracking/version` is `0.9.5` but the document status is `final`.

A tool MAY set the document status to `draft` as a quick fix.

6.1.18 Released Revision History

It MUST be tested that no item of the revision history has a `number` of `0` or `0.y.z` when the document status is `final` or `interim`.

The relevant path for this test is:

```
/document/tracking/revision_history[]/number
```

Example 1 (which fails the test):

```
"tracking": {
  // ...
  "revision_history": [
    {
      "date": "2023-09-17T10:00:00.000Z",
      "number": "0",
      "summary": "First draft"
    },
    {
      "date": "2024-01-21T10:00:00.000Z",
      "number": "1",
      "summary": "Initial version."
    }
  ],
  "status": "final",
  "version": "1"
}
```

The document status is `final` but the revision history includes an item which has `0` as value for `number`.

A tool MAY remove the items with a `number` of `0` or `0.y.z` from the revision history as a quick fix.

6.1.19 Revision History Entries for Pre-release Versions

It MUST be tested that no item of the revision history has a `number` which includes pre-release information.

The relevant path for this test is:

```
/document/tracking/revision_history[]/number
```

Example 1 (which fails the test):

```
"revision_history": [
  {
    "date": "2023-08-22T10:00:00.000Z",
    "number": "1.0.0-rc",
    "summary": "Release Candidate for initial version."
  },
  {
    "date": "2023-08-23T10:00:00.000Z",
    "number": "1.0.0",
    "summary": "Initial version."
  }
]
```

The revision history contains an item which has a `number` that indicates that this is pre-release.

A tool MAY merge each item of the revision history that has a `number` which includes pre-release information into the corresponding item which contains the same `number` without pre-release information as a quick fix. If no such item exists, the tool MAY just remove the pre-release information from the `number` instead of merging the item.

6.1.20 Non-Draft Document Version

It MUST be tested that document version does not contain a pre-release part if the document status is `final` or `interim`.

The relevant path for this test is:

```
/document/tracking/version
```

Example 1 (which fails the test):

```
"tracking": {
  // ...
  "status": "interim",
  "version": "1.0.0-alpha"
}
```

The document status is `interim` but the document version contains the pre-release part `-alpha`.

A tool MAY remove the pre-release part from the document version as a quick fix.

6.1.21 Missing Item in Revision History

It MUST be tested that items of the revision history do not omit a version number when the items are sorted ascending by `date`. As the timestamps might use different timezones, the sorting MUST take timezones into account. In the case of semantic versioning, this applies only to the Major version. It MUST also be tested that the first item in such a sorted list has either the version number 0 or 1 in the case of integer versioning or a Major version of 0 or 1 in the case of semantic versioning.

The relevant path for this test is:

```
/document/tracking/revision_history
```

Example 1 (which fails the test):

```
"revision_history": [
  {
    "date": "2023-08-22T10:00:00.000Z",
    "number": "1",
    "summary": "Initial version."
  },
  {
    "date": "2024-01-21T10:00:00.000Z",
    "number": "3",
    "summary": "Some other changes."
  }
]
```

The item for version 2 is missing.

A tool MAY add a stub for the missing item to the revision history as a quick fix. The stub might miss the `date` and `summary` which have to be provided by the user or other data sources.

6.1.22 Multiple Definition in Revision History

It MUST be tested that items of the revision history do not contain the same version number.

The relevant path for this test is:

```
/document/tracking/revision_history
```

Example 1 (which fails the test):

```
"revision_history": [
  {
    "date": "2024-01-20T10:00:00.000Z",
    "number": "1",
    "summary": "Initial version."
  },
  {
    "date": "2024-01-21T10:00:00.000Z",
    "number": "1",
    "summary": "Some other changes."
  }
]
```


The revision history contains two items with the version number `1`.

6.1.23 Multiple Use of Same CVE

It MUST be tested that a CVE is not used in multiple vulnerability items.

The relevant path for this test is:

```
/vulnerabilities[]/cve
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "cve": "CVE-2017-0145"
  },
  {
    "cve": "CVE-2017-0145"
  }
]
```

The vulnerabilities array contains two items with the same CVE identifier `CVE-2017-0145`.

6.1.24 Multiple Definition in Involvements

It MUST be tested that items of the list of involvements do not contain the same `party` regardless of its `status` more than once at any `date`.

The relevant path for this test is:

```
/vulnerabilities[]/involvements
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "involvements": [
      {
        "date": "2023-08-23T10:00:00.000Z",
        "party": "vendor",
        "status": "completed"
      },
      {
        "date": "2023-08-23T10:00:00.000Z",
        "party": "vendor",
        "status": "in_progress",
        "summary": "The vendor has released a mitigation and is working to fully resc
      }
    ]
  }
]
```


The list of involvements contains two items with the same tuple `party` and `date`.

6.1.25 Multiple Use of Same Hash Algorithm

It MUST be tested that the same hash algorithm is not used multiple times in one item of file hashes.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/hashes[]
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/product_paths[]/full_product_name/product_identification_helper/hashes[
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700",
      "product_identification_helper": {
        "hashes": [
          {
            "file_hashes": [
              {
                "algorithm": "sha256",
                "value": "026a37919b182ef7c63791e82c9645e2f897a3f0b73c7a6028c7febf62e
              },
              {
                "algorithm": "sha256",
                "value": "0a853ce2337f0608489ac596a308dc5b7b19d35a52b10bf31261586ac36
              }
            ],
            "filename": "product_a.so"
          }
        ]
      }
    }
  ]
}
```

The hash algorithm `sha256` is used two times in one item of file hashes.

6.1.26 Prohibited Document Category Name

It MUST be tested that the document category is not equal to the (case insensitive) name (without the prefix `csaf_`) or value of any other profile than “CSAF Base”. Any occurrences of dash, hyphen, minus, underscore, and white space characters are removed from the values on both sides before the case insensitive match.

The characters listed above are independent of their graphical variants. In general, all corresponding unicode characters are part of the group. For example, dash, hyphen and underscore characters include, but are not limited to:

- Em dash

- En dash
- Figure dash
- Horizontal bar
- Hyphen
- Hyphen-minus
- Non-breaking hyphen
- Low line
- Combining low line
- Fullwidth low line

This applies for both, the comparison against the name and value. Also the value MUST NOT start with the reserved prefix `csaf_` except if the value is exactly `csaf_base`.

This test does only apply for CSAF documents with the profile “CSAF Base”. Therefore, it MUST be skipped if the document category matches one of the values defined for the profile other than “CSAF Base”.

For CSAF 2.1, the test must be skipped for the following values in `/document/category`:

```
csaf_base
csaf_deprecated_security_advisory
csaf_informational_advisory
csaf_security_advisory
csaf_security_incident_response
csaf_superseded
csaf_vex
csaf_withdrawn
```

This is the only mandatory test related to the profile “CSAF Base” as the required fields SHALL be checked by validating the JSON schema.

The relevant path for this test is:

```
/document/category
```

Examples 1 (for currently prohibited values):

```
Csaf_a
CsaF_VeX
CSafDeprecatedSecurity—Advisory
csafvex
Deprecated Security Advisory
Informational Advisory
Security      Advisory
security-incident-response
Superseded
V_eX
veX
withdrawn
```

Example 2 (which fails the test):

```
"category": "Security_Incident_Response"
```


> The value `Security_Incident_Response` is the name of a profile where the space was replaced with underscores.

6.1.27 Profile Tests

This subsubsection structures the mandatory tests for the profiles. Not all tests apply for all profiles. Tests SHOULD be skipped if the document category does not match the one given in the test. Each of the following tests SHOULD be treated as they where listed similar to the other tests.

> An application MAY group these tests by profiles when providing the additional function to only run one or more selected tests. This results in one virtual test per profile.

6.1.27.1 Document Notes

It MUST be tested that at least one item in `/document/notes` exists which has a `category` of `description`, `details`, `general` or `summary`.

The relevant values for `/document/category` are:

```
csaf_informational_advisory
csaf_security_incident_response
```

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"notes": [
  {
    "category": "legal_disclaimer",
    "text": "The CSAF document is provided to You \"AS IS\" and \"AS AVAILABLE\" and
    "title": "Terms of Use"
  }
]
```

> The document notes do not contain an item which has a `category` of `description`, `details`, `general` or `summary`.

6.1.27.2 Document References

It MUST be tested that at least one item in `/document/references` exists that has links to an `external` source.

The relevant values for `/document/category` are:

```
csaf_informational_advisory
csaf_security_incident_response
```

The relevant path for this test is:

```
/document/references
```


Example 1 (which fails the test):

```
  "references": [
    {
      "category": "self",
      "summary": "The canonical URL.",
      "url": "https://example.com/security/data/csaf/2024/oasis_csaf_tc-csaf_2_1-2024-
6-1-27-02-01.json"
    }
  ]
```

The document references do not contain any item which has the category `external`.

6.1.27.3 Vulnerabilities

It MUST be tested that the element `/vulnerabilities` does not exist.

The relevant values for `/document/category` are:

```
csaf_informational_advisory
csaf_withdrawn
csaf_superseded
```

The relevant path for this test is:

```
/vulnerabilities
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "title": "A vulnerability item that SHALL NOT exist"
  }
]
```

The element `/vulnerabilities` exists.

A tool MAY change the `/document/category` to `csaf_base` as a quick fix.

6.1.27.4 Product Tree

It MUST be tested that the element `/product_tree` exists.

The relevant values for `/document/category` are:

```
csaf_security_advisory
csaf_vex
csaf_deprecated_security_advisory
```

The relevant path for this test is:


```
/product_tree
```

Example 1 (which fails the test):

```
{
  "document": {
    // ...
  },
  "vulnerabilities": [
    // ...
  ]
}
```

The element `/product_tree` does not exist.

6.1.27.5 Vulnerability Notes

For each item in `/vulnerabilities` it MUST be tested that the element `notes` exists.

The relevant values for `/document/category` are:

```
csaf_security_advisory
csaf_vex
csaf_deprecated_security_advisory
```

The relevant path for this test is:

```
/vulnerabilities[]/notes
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "product_status": {
      "known_affected": [
        "CSAFPID-9080700"
      ]
    },
    "title": "A vulnerability item without a note"
  }
]
```

The vulnerability item has no `notes` element.

6.1.27.6 Product Status

For each item in `/vulnerabilities` it MUST be tested that the element `product_status` exists.

The relevant values for `/document/category` are:

```
csaf_security_advisory
csaf_deprecated_security_advisory
```

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "title": "A vulnerability item without a product status"
  }
]
```

The vulnerability item has no `product_status` element.

6.1.27.7 VEX Product Status

For each item in `/vulnerabilities` it MUST be tested that at least one of the elements `fixed`, `known_affected`, `known_not_affected`, or `under_investigation` is present in `product_status`.

The relevant value for `/document/category` is:

```
csaf_vex
```

The relevant paths for this test are:

```
/vulnerabilities[]/product_status/fixed
/vulnerabilities[]/product_status/known_affected
/vulnerabilities[]/product_status/known_not_affected
/vulnerabilities[]/product_status/under_investigation
```

Example 1 (which fails the test):

```
"product_status": {
  "first_fixed": [
    // ...
  ],
  "recommended": [
    // ...
  ]
}
```


None of the elements `fixed`, `known_affected`, `known_not_affected`, or `under_investigation` is present in `product_status`.

6.1.27.8 Vulnerability ID

For each item in `/vulnerabilities` it MUST be tested that at least one of the elements `cve` or `ids` is present.

The relevant value for `/document/category` is:

```
csaf_vex
```

The relevant paths for this test are:

```
/vulnerabilities[]/cve
/vulnerabilities[]/ids
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "title": "A vulnerability item without a CVE or ID"
  }
]
```

None of the elements `cve` or `ids` is present.

6.1.27.9 Impact Statement

For each item in `/vulnerabilities[]/product_status/known_not_affected` it MUST be tested that a corresponding impact statement exist in `/vulnerabilities[]/flags` or `/vulnerabilities[]/threats`. For the latter one, the `category` value for such a statement MUST be `impact`.

The relevant value for `/document/category` is:

```
csaf_vex
```

The relevant path for this test is:

```
/vulnerabilities[]/flags
/vulnerabilities[]/threats
```

Example 1 (which fails the test):


```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    },
    {
      "product_id": "CSAFPID-9080701",
      "name": "Product B"
    },
    {
      "product_id": "CSAFPID-9080702",
      "name": "Product C"
    }
  ],
  "product_groups": [
    {
      "group_id": "CSAFGID-0001",
      "product_ids": [
        "CSAFPID-9080700",
        "CSAFPID-9080701"
      ]
    }
  ]
},
"vulnerabilities": [
  {
    // ...
    "product_status": {
      "known_not_affected": [
        "CSAFPID-9080700",
        "CSAFPID-9080701",
        "CSAFPID-9080702"
      ]
    },
    "threats": [
      {
        "category": "impact",
        "details": "The vulnerable code is not present in these products.",
        "group_ids": [
          "CSAFGID-0001"
        ]
      }
    ]
  }
]
```

There is no impact statement for CSAFPID-9080702.
Note: The impact statement for CSAFPID-9080700 and CSAFPID-9080701 is given through CSAFGID-0001.

6.1.27.10 Action Statement

For each item in `/vulnerabilities[]/product_status/known_affected` it MUST be tested that a corresponding action statement exist in `/vulnerabilities[]/remediations`.

The relevant value for `/document/category` is:

```
csaf_vex
```

The relevant path for this test is:

```
/vulnerabilities[]/remediations
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    },
    {
      "product_id": "CSAFPID-9080701",
      "name": "Product B"
    },
    {
      "product_id": "CSAFPID-9080702",
      "name": "Product C"
    }
  ],
  "product_groups": [
    {
      "group_id": "CSAFGID-0001",
      "product_ids": [
        "CSAFPID-9080700",
        "CSAFPID-9080701"
      ],
      "summary": "EOL products"
    }
  ]
},
"vulnerabilities": [
  {
    // ...
    "product_status": {
      "known_affected": [
        "CSAFPID-9080700",
        "CSAFPID-9080701",
        "CSAFPID-9080702"
      ]
    },
    "remediations": [
```



```
        {
          "category": "no_fix_planned",
          "details": "These products are end-of-life. Therefore, no fix will be provide
          "group_ids": [
            "CSAFGID-0001"
          ]
        }
      ]
    }
  ]
```

There is no action statement for CSAFPID-9080702.
Note: The action statement for CSAFPID-9080700 and CSAFPID-9080701 is given through CSAFGID-0001.

6.1.27.11 Vulnerabilities

It MUST be tested that the element /vulnerabilities exists.

The relevant values for /document/category are:

```
csaf_security_advisory
csaf_vex
csaf_deprecated_security_advisory
```

The relevant path for this test is:

```
/vulnerabilities
```

Example 1 (which fails the test):

```
{
  "document": {
    // ...
  },
  "product_tree": [
    // ...
  ]
}
```

The element /vulnerabilities does not exist.

6.1.27.12 Affected Products

For each item in /vulnerabilities it MUST be tested that the element product_status/known_affected exists.

The relevant value for /document/category is:

```
csaf_security_advisory
```

The relevant path for this test is:


```
/vulnerabilities[]/product_status/known_affected
```

Example 1 (which fails the test):

```
"product_status": {
  "under_investigation": [
    "CSAFPID-9080700"
  ]
}
```

The product status does not contain the `known_affected` element.

6.1.27.13 Corresponding Affected Products

For each product listed in the product status group fixed in any vulnerability, it MUST be tested that a corresponding version of the product is listed as affected in the same vulnerability.

For a product path including the `installed_with` relationship the product path leading to but not including the relationship is a corresponding product. Such product path could also be just the product identified by `beginning_product_reference` if the first subpath element has the category `installed_with`.

The relevant value for `/document/category` is:

```
csaf_security_advisory
```

The relevant path for this test is:

```
/vulnerabilities[]/product_status/known_affected
```

Example 1 (which fails the test):

```
{
  // ...
  "product_tree": {
    "branches": [
      {
        "branches": [
          {
            "branches": [
              {
                "category": "product_version",
                "name": "4.2",
                "product": {
                  "name": "Example Company Product A 4.2",
                  "product_id": "CSAFPID-9080700"
                }
              }
            ],
            "category": "product_name",
            "name": "Product A"
          }
```



```
        ],
        "category": "vendor",
        "name": "Example Company"
      }
    ]
  },
  "vulnerabilities": [
    {
      // ...
      "product_status": {
        "fixed": [
          "CSAFPID-9080700"
        ]
      }
    }
  ]
}
```

The vulnerability just contains the fixed product but does not list corresponding affected products.

6.1.27.14 Document Notes

It MUST be tested that at least one item in `/document/notes` exists which has a `category` of `description`.

The relevant values for `/document/category` are:

```
csaf_withdrawn
csaf_superseded
```

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"notes": [
  {
    "category": "legal_disclaimer",
    "text": "The CSAF document is provided to You \"AS IS\" and \"AS AVAILABLE\" and
    "title": "Terms of Use"
  }
]
```

The document notes do not contain an item which has a `category` of `description`.

6.1.27.15 Product Tree

It MUST be tested that the element `/product_tree` does not exist.

The relevant values for `/document/category` are:

```
csaf_withdrawn
csaf_superseded
```

The relevant path for this test is:

```
/product_tree
```

Example 1 (which fails the test):

```
"product_tree": [
  // ...
]
```

The element `/product_tree` exists.

6.1.27.16 Revision History

It MUST be tested that the revision history contains at least two entries.

The relevant values for `/document/category` are:

```
csaf_withdrawn
csaf_superseded
```

The relevant path for this test is:

```
/document/tracking/revision_history
```

Example 1 (which fails the test):

```
"revision_history": [
  {
    "date": "2024-01-24T10:00:00.000Z",
    "number": "1",
    "summary": "Initial version."
  }
],
```

The revision history contains only one entry.

6.1.27.17 Reasoning for Withdrawal

If the document language is English or unspecified, it MUST be tested that exactly one item in document notes exists that has the title `Reasoning for Withdrawal`. The `category` of this item MUST be `description`.

The relevant value for `/document/category` is:

```
csaf_withdrawn
```

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"notes": [
  {
    "category": "summary",
    "text": "This CSAF document contained example data and was withdrawn to create te
    "title": "Reasoning for Withdrawal"
  }
],
```

The note has the correct title. However, it uses the wrong category.

6.1.27.18 Reasoning for Supersession

If the document language is English or unspecified, it MUST be tested that exactly one item in document notes exists that has the title `Reasoning for Supersession`. The `category` of this item MUST be `description`.

The relevant value for `/document/category` is:

```
csaf_superseded
```

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"notes": [
  {
    "category": "details",
    "text": "This CSAF document contained example data and was withdrawn to create te
    "title": "Reasoning for Supersession"
  }
],
```

The note has the correct title. However, it uses the wrong category.

6.1.27.19 Reference to Superseding Document

If the document language is English or unspecified, it MUST be tested that at least one item in document references exists that has a summary starting with `Superseding Document`. The `category` of this item MUST be `external`.

The relevant value for `/document/category` is:

```
csaf_superseded
```

The relevant path for this test is:

```
/document/references
```

Example 1 (which fails the test):

```
"references": [
  {
    "category": "self",
    "summary": "Superseding Document",
    "url": "https://example.com/.well-known/csaf/clear/2024/esa-2024-1234.json"
  }
],
```

The reference summary starts correctly with the string “Superseding Document”. However, it uses the wrong category.

6.1.28 Translation

It MUST be tested that the given source language and document language are not the same.

The relevant path for this test is:

```
/document/lang
/document/source_lang
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "lang": "en-US",
  // ...
  "source_lang": "en-US",
  // ...
}
```

The document language and the source language have the same value `en-US`.
Note: A translation from `en-US` to `en-GB` would pass the test.

A tool MAY remove the source language as a quick fix.

6.1.29 Remediation without Product Reference

For each item in `/vulnerabilities[]/remediations` it MUST be tested that it includes at least one of the elements `group_ids` or `product_ids`.

The relevant path for this test is:

```
/vulnerabilities[]/remediations[]
```

Example 1 (which fails the test):

```
"remediations": [
  {
    "category": "no_fix_planned",
    "details": "These products are end-of-life. Therefore, no fix will be provide
  }
]
```

The given remediation does not specify to which products it should be applied.

A tool MAY add all products of the affected group of this vulnerability to the remediation as a quick fix.

6.1.30 Mixed Integer and Semantic Versioning

It MUST be tested that all elements of type `/$defs/version_t` follow either integer versioning or semantic versioning homogeneously within the same document.

The relevant paths for this test are:

```
/document/tracking/revision_history[]/number
/document/tracking/version
```

Example 1 (which fails the test):

```
"tracking": {
  // ...
  "revision_history": [
    {
      "date": "2024-01-21T09:00:00.000Z",
      "number": "1.0.0",
      "summary": "Initial version."
    },
    {
      "date": "2024-01-21T10:00:00.000Z",
      "number": "2",
      "summary": "Second version."
    }
  ],
  // ...
  "version": "2"
}
```

The document started with semantic versioning (`1.0.0`) and switched to integer versioning (`2`).

A tool MAY assign all items their corresponding value according to integer versioning as a quick fix. In such case, the old `number` SHOULD be stored in `legacy_version`.

6.1.31 Version Range in Product Version

For each element of type `/$defs/branches_t` with `category` of `product_version` it MUST be tested that the value of `name` does not contain a version range.

To implement this test it is deemed sufficient that, when converted to lowercase, the value of `name` satisfies the two requirements below:

1. It does not contain any of the following operators:

   ```
   <
   <=
   >
   >=
   ```

2. If interpreted as a list of individual words separated by white space, the list does not contain any of the following keywords:

   ```
   after
   all
   before
   earlier
   later
   prior
   versions
   ```

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
"branches": [
  {
    "category": "product_version",
    "name": "prior to 4.2",
    // ...
  }
]
```

The version range `prior to 4.2` is given for the branch category `product_version`.

6.1.32 Flag without Product Reference

For each item in `/vulnerabilities[]/flags` it MUST be tested that it includes at least one of the elements `group_ids` or `product_ids`.

The relevant path for this test is:

`/vulnerabilities[]/flags[]`

Example 1 (which fails the test):

```
"flags": [
  {
    "label": "component_not_present"
  }
]
```

The given flag does not specify to which products it should be applied.

6.1.33 Multiple Flags with VEX Justification Codes per Product

For each item in `/vulnerabilities[]` it MUST be tested that a Product is not member of more than one Flag item with a VEX justification code (see section 3.2.4.7). This takes indirect relations through Product Groups into account.

Additional flags with a different purpose might be provided in later versions of CSAF. Through the explicit reference of VEX justification codes the test is specified to be forward-compatible.

The relevant path for this test is:

`/vulnerabilities[]/flags`

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    },
    {
      "product_id": "CSAFPID-9080701",
      "name": "Product B"
    }
  ],
  "product_groups": [
    {
      "group_id": "CSAFGID-0001",
      "product_ids": [
        "CSAFPID-9080700",
        "CSAFPID-9080701"
      ]
    }
  ]
},
"vulnerabilities": [
  {
    // ...
    "flags": [
```



```
        {
          "label": "component_not_present",
          "group_ids": [
            "CSAFGID-0001"
          ]
        },
        {
          "label": "vulnerable_code_cannot_be_controlled_by_adversary",
          "product_ids": [
            "CSAFPID-9080700"
          ]
        }
      ],
      // ...
      "product_status": {
        "known_not_affected": [
          "CSAFPID-9080700",
          "CSAFPID-9080701"
        ]
      }
    }
  ]
```

There are two flags given for `CSAFPID-9080700` - one indirect through `CSAFGID-0001` and one direct.

6.1.34 Branches Recursion Depth

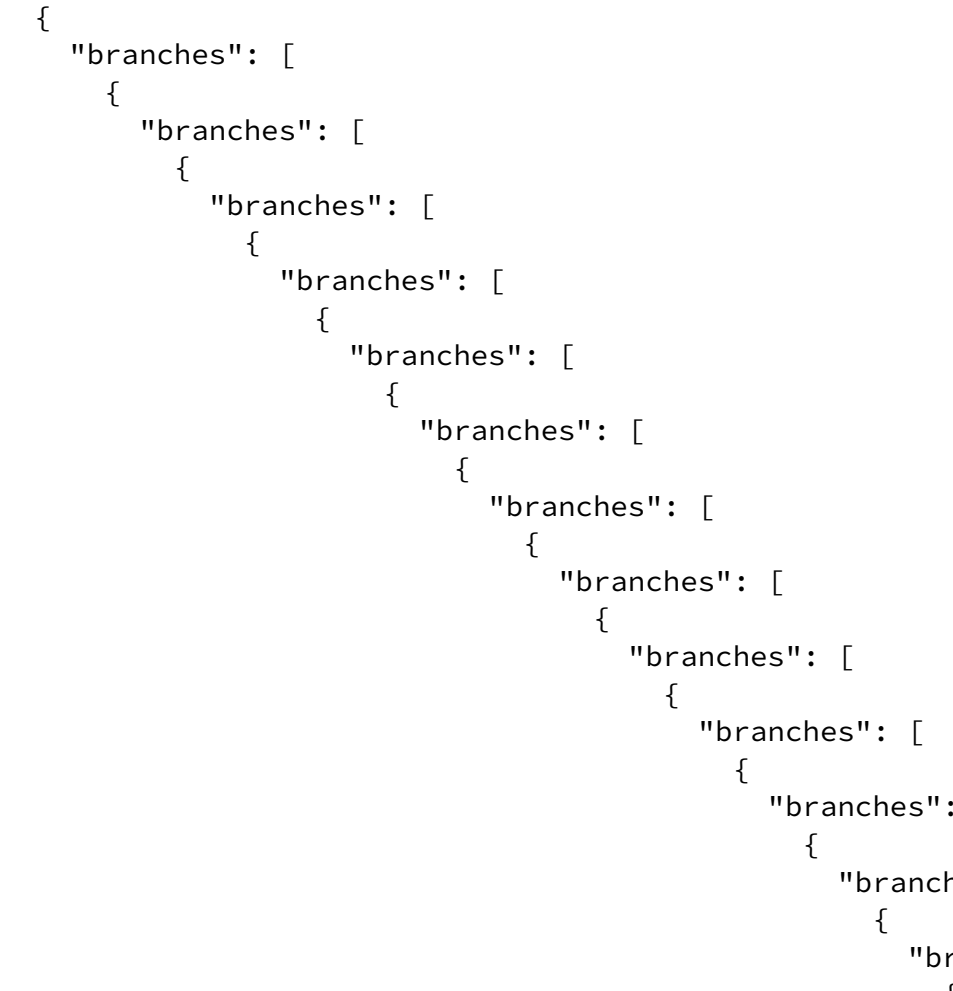
For each product defined under `/product_tree/branches[]` it MUST be tested that the complete JSON path does not contain more than 30 instances of `branches`.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/product
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "branches": [
                {
                  "branches": [
                    {
                      "branches": [
                        {
                          "branches": [
                            {
                              "branches": [
                                {
                                  "branches": [
```

9080700"


```
                                                                    }
                                                                  ],
                                                                  "category": "pr
                                                                  "name": "less"
                                                                }
                                                              ],
                                                              "category": "produc
                                                              "name": "or"
                                                            }
                                                          ],
                                                          "category": "product_fa
                                                          "name": "more"
                                                        }
                                                      ],
                                                      "category": "product_family
                                                      "name": "and"
                                                    }
                                                  ],
                                                  "category": "product_family",
                                                  "name": "unnecessary"
                                                }
                                              ],
                                              "category": "product_family",
                                              "name": "seem"
                                            }
                                          ],
                                          "category": "product_family",
                                          "name": "which"
                                        }
                                      ],
                                      "category": "product_family",
                                      "name": "product"
                                    }
                                  ],
                                  "category": "product_family",
                                  "name": "hypothetical"
                                }
                              ],
                              "category": "product_family",
                              "name": "this"
                            }
                          ],
                          "category": "product_family",
                          "name": "for"
                        }
                      ],
                      "category": "product_family",
                      "name": "structure"
                    }
                  ],
                  "category": "product_family",
                  "name": "nested"
                }
```



```
                ],
                "category": "product_family",
                "name": "deeply"
              }
            ],
            "category": "product_family",
            "name": "a"
          }
        ],
        "category": "product_family",
        "name": "uses"
      }
    ],
    "category": "vendor",
    "name": "Example Company"
  }
 ]
}
```

The complete JSON path contains 31 times `branches`.

6.1.35 Contradicting Remediations

For each item in `/vulnerabilities[]/remediations` it MUST be tested that a product is not member of contradicting remediation categories (see table tab). This takes indirect relations through product groups into account.

The relevant path for this test is:

```
/vulnerabilities[]/remediations[]
```

Example 1 (which fails the test):

```
"remediations": [
  {
    "category": "no_fix_planned",
    "details": "The product is end-of-life. Therefore, no fix will be provided.",
    "product_ids": [
      "CSAFPID-9080700"
    ]
  },
  {
    "category": "vendor_fix",
    "details": "Update to version >=14.3 to fix the vulnerability.",
    "product_ids": [
      "CSAFPID-9080700"
    ]
  }
]
```

The two remediations given for the product with product ID `CSAFPID-908070` contradict each other.

A tool MAY apply the conversion rules from the conformance target CSAF 2.0 to CSAF 2.1 converter if applicable or remove the product from the remediation with the lower priority. The priority MAY be defined as follows: `vendor_fix` > `mitigation` > `workaround` > `fix_planned` > `no_fix_planned` > `optional_patch` > `none_available`

6.1.36 Contradicting Product Status Remediation Combination

For each item in `/vulnerabilities[]/remediations` it MUST be tested that a product is not member of a contradicting product status group (see table tab). This takes indirect relations through product groups into account.

The relevant path for this test is:

```
/vulnerabilities[]/remediations[]
```

Example 1 (which fails the test):

```
"product_status": {
  "known_not_affected": [
    "CSAFPID-9080700"
  ]
},
"remediations": [
  {
    "category": "vendor_fix",
    "details": "Update to version >=14.3 to fix the vulnerability.",
    "product_ids": [
      "CSAFPID-9080700"
    ]
  }
]
```

For the product with product ID `CSAFPID-908070` a `vendor_fix` is given but the product was not affected at all.

6.1.37 Date and Time

For each item of type `string` and format `date-time` it MUST be tested that it conforms to the rules given in section 2.3.

The relevant path for this test is:

```
/document/tracking/current_release_date
/document/tracking/generator/date
/document/tracking/initial_release_date
/document/tracking/revision_history[]/date
/vulnerabilities[]/disclosure_date
/vulnerabilities[]/discovery_date
/vulnerabilities[]/first_known_exploitation_dates[]/date
/vulnerabilities[]/first_known_exploitation_dates[]/exploitation_date
/vulnerabilities[]/flags[]/date
/vulnerabilities[]/involvements[]/date
/vulnerabilities[]/remediations[]/date
/vulnerabilities[]/threats[]/date
```


Example 1 (which fails the test):

```
  "current_release_date": "2024-01-24 10:00:00.000Z",
```

The `current_release_date` uses a white space as separator instead the letter `T`.

6.1.38 Non-Public Sharing Group with Max UUID

It MUST be tested that a CSAF document using Max UUID as sharing group ID has the TLP label `CLEAR`.

The relevant path for this test is:

```
/document/distribution/tlp/label
```

Example 1 (which fails the test):

```
"distribution": {
  "sharing_group": {
    "id": "ffffffff-ffff-ffff-ffff-ffffffffffff",
    "name": "Public"
  },
  "tlp": {
    "label": "RED"
  }
},
```

The sharing group uses the Max UUID but the CSAF document is labeled as `TLP:RED`.

A tool MAY remove the property `sharing_group` as a quick fix.

6.1.39 Public Sharing Group with No Max UUID

It MUST be tested that a CSAF document with the TLP label `CLEAR` use the Max UUID as sharing group ID if any. The test SHALL pass if no sharing group is present or the Nil UUID is used and the document status is `draft`.

The relevant path for this test is:

```
/document/distribution/sharing_group/id
```

Example 1 (which fails the test):

```
"distribution": {
  "sharing_group": {
    "id": "5868d6be-b28a-404e-a245-0b5093b31b8b"
  },
  "tlp": {
    "label": "CLEAR"
  }
},
```


The sharing group is present for the `TLP:CLEAR` document but it differs from the Max UUID.

A tool MAY update the sharing group id as a quick fix.

6.1.40 Invalid Sharing Group Name

It MUST be tested that the value of sharing group name does not equal the reserved values from section 3.2.2.5.1 if the precondition is not fulfilled.

The relevant path for this test is:

```
/document/distribution/sharing_group/name
```

Example 1 (which fails the test):

```
"distribution": {
  "sharing_group": {
    "id": "5868d6be-b28a-404e-a245-0b5093b31b8b",
    "name": "Public"
  },
  // ...
},
```

The sharing group name is `Public` but it does not use the Max UUID.

A tool MAY update the sharing group name as a quick fix.

6.1.41 Missing Sharing Group Name

It MUST be tested that the sharing group name exists and equals the predefined reserved value from section 3.2.2.5.1 if the precondition is fulfilled.

The relevant path for this test is:

```
/document/distribution/sharing_group/name
```

Example 1 (which fails the test):

```
"distribution": {
  "sharing_group": {
    "id": "ffffffff-ffff-ffff-ffff-ffffffffffff"
  },
  // ...
},
```

The Max UUID is used but the sharing group name does not exist.

A tool MAY add the corresponding sharing group name as a quick fix.

6.1.42 PURL Qualifiers

For each `product_identification_helper` object containing multiple PURLs it MUST be tested that the PURLs only differ in their qualifiers.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/purls[]
/product_tree/full_product_names[]/product_identification_helper/purls[]
/product_tree/product_paths[]/full_product_name/product_identification_helper/purls[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700",
      "product_identification_helper": {
        "purls": [
          "pkg:maven/com.example/logging@1.3.4",
          "pkg:maven/com.example/audit@1.3.4"
        ]
      }
    }
  ]
}
```

The two PURLs differ in the name component.

6.1.43 Use of Multiple Stars in Model Number

For each model number it MUST be tested that the it does not contain multiple unescaped stars.

Multiple `*` that match zero or multiple characters within a model number introduce ambiguity and are therefore prohibited.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/model_nu
/product_tree/full_product_names[]/product_identification_helper/model_numbers[]
/product_tree/product_paths[]/full_product_name/product_identification_helper/model_n
```

Example 1 (which fails the test):

```
"model_numbers": [
  "P*A*"
]
```

The model number contains two unescaped stars.

6.1.44 Use of Multiple Stars in Serial Number

For each serial number it MUST be tested that the it does not contain multiple unescaped stars.

> Multiple `*` that match zero or multiple characters within a serial number introduce ambiguity and are therefore prohibited.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/serial_r
/product_tree/full_product_names[]/product_identification_helper/serial_numbers[]
/product_tree/product_paths[]/full_product_name/product_identification_helper/serial_
```

Example 1 (which fails the test):

```
"serial_numbers": [
  "P*A*"
]
```

> The serial number contains two unescaped stars.

6.1.45 Inconsistent Disclosure Date

For each vulnerability, it MUST be tested that the `disclosure_date` is earlier than or equal to the `date` of the newest item of the `revision_history` if the document is labeled `TLP:CLEAR` and the document status is `final` or `interim`. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The relevant path for this test is:

```
/vulnerabilities[]/disclosure_date
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "distribution": {
    "tlp": {
      "label": "CLEAR"
    }
  },
  // ...
  "tracking": {
    // ...
    "revision_history": [
      {
        "date": "2024-01-24T10:00:00.000Z",
        "number": "1",
        "summary": "Initial version."
      }
    ],
    "status": "final",
    // ...
```



```
    }
  },
  "vulnerabilities": [
    {
      "disclosure_date": "2024-02-24T10:00:00.000Z"
    }
  ]
```

The document is labeled `TLP:CLEAR` and in status `final` but the `disclosure_date` is newer than the `date` of newest item in the `revision_history`.

6.1.46 Invalid SSVC

It MUST be tested that the given SSVC object is valid according to the referenced schema.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2
```

Example 1 (which fails the test):

```
"ssvc_v2": {
  "schemaVersion": "2.0.0",
  "timestamp": "2024-01-24T10:00:00.000Z"
}
```

The required element `selections` is missing.

6.1.47 Inconsistent SSVC Target IDs

For each `ssvc_v2` object it MUST be tested that each item in `target_ids` is either the CVE of the vulnerability given in `cve` or the `text` of an item in the `ids` array of the vulnerability. The test MUST fail, if the target ID equals the `/document/tracking/id` and the CSAF document contains more than one vulnerability.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/target_ids[]
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "cve": "CVE-1900-0001",
    "metrics": [
      {
        "content": {
          "ssvc_v2": {
            // ...
            "target_ids": [
              "CVE-1900-0002"
            ],
```



```
            // ...
          }
        },
        // ...
      }
    ]
  }
]
```

The SSVC Target ID does not match the CVE ID.

A tool MAY remove any inconsistent SSVC Target ID as a quick fix.

If the tool removes the last item of the array, it MUST remove the element `target_ids` as well.

6.1.48 SSVC Decision Points

For each SSVC decision point given under `selections` within a registered base `namespace`, it MUST be tested that given decision point exists, is valid and the items in `values` are ordered correctly. The test SHALL pass, if a unregistered `namespace` is used. Namespaces reserved for special purpose MUST be treated as per their definition.

A list of all currently registered namespaces is a available in the SSVC documentation at [SSVC-RNS].
A list of all valid decision points of registered namespaces including their values is available at the SSVC repository (see [SSVC-DP]). The items in `values` need to have the same order as in their definition.
This test also covers decision points which use a registered base namespace with any number of extensions. The rules for namespace extensions as specified by SSVC must be applied in such case. Note that:

- Extensions can limit the number of available values, but not add new ones or change the order.
- Extensions can translate or refine the `name` and `definition`, but not change the `key`.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]
```

Example 1 (which fails the test):

```
"ssvc_v2": {
  // ...
  "selections": [
    {
      "key": "MI",
      "namespace": "ssvc",
      "values": [
        {
          "key": "N",
          "name": "None"
        },
        {
          "key": "D",
          "name": "Degraded"
        }
      ],
```



```
        "version": "1.0.0"
      }
    ],
    // ...
  }
```

The SSVC decision point `Mission Impact` doesn't have the value `Degraded` in version `1.0.0`.

If applicable, a tool MAY sort the items in `values` according to the order of their definition as a quick fix.

6.1.49 Inconsistent SSVC Timestamp

For each vulnerability, it MUST be tested that the each SSVC `timestamp` is earlier than or equal to the `date` of the newest item of the `revision_history` if the document status is `final` or `interim`. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/timestamp
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "tracking": {
    // ...
    "revision_history": [
      {
        "date": "2024-01-24T10:00:00.000Z",
        "number": "1",
        "summary": "Initial version."
      }
    ],
    "status": "final",
    // ...
  }
},
// ...
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "ssvc_v2": {
            // ...
            "timestamp": "2024-07-13T10:00:00.000Z"
          }
        },
        // ...
      }
    ]
  }
]
```


The document is in status `final` but the SSVC `timestamp` is newer than the `date` of newest item in the `revision_history`.

6.1.50 Product Version Range Rules

For each element of type `/$defs/branches_t` with `category` of `product_version_range`, it MUST be tested that the value of `name` complies with the rules given in section 3.1.2.3.2. Version schemes of vers not supported by the implementation SHALL result in a warning which MUST include the version scheme name used. Nevertheless, all other rules MUST be checked to the extend possible.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
{
  "category": "product_version_range",
  "name": "all versions < 4.2.0",
  // ...
}
```

The version range given does not comply with the rules given in section 3.1.2.3.2.

6.1.51 Inconsistent EPSS Timestamp

For each vulnerability, it MUST be tested that the each EPSS `timestamp` is earlier than or equal to the `date` of the newest item of the `revision_history` if the document status is `final` or `interim`. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/epss/timestamp
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "distribution": {
    "tlp": {
      "label": "CLEAR"
    }
  },
  // ...
  "tracking": {
    // ...
    "revision_history": [
      {
        "date": "2024-01-24T10:00:00.000Z",
        "number": "1",
        "summary": "Initial version."
```



```
        }
      ],
      "status": "final",
      // ...
    }
  },
  "vulnerabilities": [
    {
      "cve": "CVE-1900-0001",
      "metrics": [
        {
          "content": {
            "epss": {
              "percentile": "0.999990000",
              "probability": "0.975570000",
              "timestamp": "2024-07-13T10:00:00.000Z"
            }
          },
          "products": [
            "CSAFPID-9080700"
          ]
        }
      ]
    }
  ]
```

The document is in status `final` but the EPSS `timestamp` is newer than the `date` of newest item in the `revision_history`.

6.1.52 Inconsistent First Known Exploitation Dates

For each First Known Exploitation Dates item, it MUST be tested that the values of its `date` and `exploitation_date` properties are both earlier than or equal to the `date` of the newest item of the `revision_history` if the document status is `final` or `interim`. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The relevant paths for this test are:

```
/vulnerabilities[]/first_known_exploitation_dates[]/date
/vulnerabilities[]/first_known_exploitation_dates[]/exploitation_date
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "tracking": {
    // ...
    "revision_history": [
      {
        "date": "2024-01-24T10:00:00.000Z",
        "number": "1",
        "summary": "Initial version."
```



```
        }
      ],
      "status": "final",
      // ...
    }
  },
  // ...
  "vulnerabilities": [
    {
      "first_known_exploitation_dates": [
        {
          "date": "2024-01-24T13:00:00.000Z",
          "exploitation_date": "2024-01-24T12:34:56.789Z",
          // ...
        }
      ]
    }
  ]
```

The document is in status `final` but the values of the properties `date` and `exploitation_date` of the First Known Exploitation Dates item are newer than the `date` of newest item in the `revision_history`.

6.1.53 Inconsistent Exploitation Date

For each First Known Exploitation Dates item, it MUST be tested that the value of `exploitation_date` is earlier than or equal to value of the sibling element `date`. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The relevant path for this test is:

```
/vulnerabilities[]/first_known_exploitation_dates[]/exploitation_date
```

Example 1 (which fails the test):

```
"first_known_exploitation_dates": [
  {
    "date": "2024-01-24T10:00:00.000Z",
    "exploitation_date": "2024-02-25T11:00:22.987Z",
    // ...
  }
]
```

The value of `exploitation_date` is newer than the value of `date` in the same item.

6.1.54 License Expression

It MUST be tested that the license expression is valid.

To implement this test, it it deemed sufficient to check for the ABNF defined in annex B of [SPDX301] and the restriction on the `DocumentRef` part given in 3.2.2.7.

The relevant path for this test is:


```
/document/license_expression
```

Example 1 (which fails the test):

```
"license_expression": "This is a license text that should not be here.",
```

The license expression contains a license text instead of a SPDX license expression.

6.1.55 License Text

If the document language is English or unspecified, and the `license_expression` contains license identifiers or exceptions that are not listed in the SPDX license list or AboutCode's "ScanCode LicenseDB", it MUST be tested that exactly one item in document notes exists that has the title `License`. The category of this item MUST be `legal_disclaimer`.

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "license_expression": "LicenseRef-www.example.com-no-work-pd",
  "notes": [
    {
      "category": "other",
      "text": "This is not a work and therefore it can't be licensed. Use it as publi
      "title": "License"
    }
  ],
  // ...
}
```

The note has the correct title. However, it uses the wrong category.

6.1.56 Use of CVSS and Qualitative Severity Rating

For each item in `/vulnerabilities` it MUST be tested that no Qualitative Severity Rating and CVSS values are listed for the tuple of Product ID and source.

Different source might assign different metrics for the same product.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]
```

Example 1 (which fails the test):


```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "cvss_v3": {
            // ...
          }
        },
        "products": [
          "CSAFPID-9080700"
        ]
      },
      {
        "content": {
          "qualitative_severity_rating": "critical"
        },
        "products": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

A CVSS v3.1 score and a Qualitative Severity Rating is given for `CSAFPID-9080700` by the document author.

A tool MAY remove the `qualitative_severity_rating` as a quick fix.

6.1.57 Stacked Branch Categories

For each `full_product_name_t` element under `/product_tree/branches`, it MUST be tested that branch categories used along the path leading to the `full_product_name_t` element appear only once in the path. The sole exception is `product_family` which can occur multiple times. Therefore, `product_family` MUST be ignored in the test.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/category
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
```



```
      {
        "branches": [
          {
            // ...
            "category": "vendor",
            "name": "ISDuBA"
          }
        ],
        "category": "vendor",
        "name": "Open Source"
      }
    ]
  }
```

The category `vendor` was used twice along the path.

A tool MAY collapse the stacked branch categories as a quick fix, if applicable.

6.1.58 Use of `product_version` in one Path with `product_version_range`

For each `full_product_name_t` element under `/product_tree/branches`, it MUST be tested that only one of the branch categories `product_version` and `product_version_range` is used along the path leading to the `full_product_name_t` element.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/category
```

Example 1 (which fails the test):

```
{
  "branches": [
    {
      "category": "product_version_range",
      "name": "vers:intdot/<1.1.1",
      "product": {
        // ...
      }
    }
  ],
  "category": "product_version",
  "name": "1.1"
},
```

Both categories `product_version` and `product_version_range` were used along the path.

A tool MAY convert the `product_version` into the `product_name` element which contains the value of the `product_version` appended to the original product name as a quick fix, if applicable. In such conversion, a tool MAY convert a previously existing `product_name` element into a `product_family`, if applicable.

6.1.59 Single Version as Product Version Range

For each element of type `/$defs/branches_t` with `category` of `product_version_range`, it MUST be tested that the value of `name` does not identify only a single version.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
{
  "category": "product_version_range",
  "name": "vers:intdot/4.2.0",
  // ...
}
```

The version range given identifies just a single version.

A tool MAY change the branch category to `product_version` and replace the value for `name` with the version as a quick fix.

6.1.60 Extension Tests

This subsubsection structures the mandatory tests for extensions. Each of the following tests SHOULD be treated as they where listed similar to the other tests.

An application MAY group these tests when providing the additional function to only run one or more selected tests.

6.1.60.1 Content Schema

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that the item is valid against the Extension Content Schema.

The relevant paths for this test are:

```
/document/x_extensions[]
/product_tree/branches[](/branches[])*/product/x_extensions[]
/product_tree/full_product_names[]/x_extensions[]
/product_tree/product_paths[]/full_product_name/x_extensions[]
/vulnerabilities[]/metrics[]/content/x_extensions[]
/vulnerabilities[]/x_extensions[]
/x_extensions[]
```

Example 1 (which fails the test):

```
"x_extensions": [
  {
    "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/cs
01/documentation-01-content_1.0.0.json",
    "category": "high_value",
    "content": {
```



```
        "documentation": "This extension is for documentation and test purposed only. I
      }
    }
  ]
```

The extension is missing the required property `critical`.

6.1.60.2 Extension Schema

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that the item is valid against the declared CSAF Extension Schema. CSAF Extensions not supported by the implementation SHALL result in a warning which MUST include the value of `$schema` of the extension. Such warning SHALL differentiate between the different classes of extensions.

The relevant paths for this test are:

```
/document/x_extensions[]
/product_tree/branches[](/branches[])*/product/x_extensions[]
/product_tree/full_product_names[]/x_extensions[]
/product_tree/product_paths[]/full_product_name/x_extensions[]
/vulnerabilities[]/metrics[]/content/x_extensions[]
/vulnerabilities[]/x_extensions[]
/x_extensions[]
```

Example 1 (which fails the test):

```
"x_extensions": [
  {
    "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/cs
12/documentation-12-content_1.0.0.json",
    // ...
    "content": {
      "notes": [
        "This instance is invalid even though it validates against the CSAF Content S
        "It misses the property `documentation` as required by the declared schema."
      ]
    },
    // ...
  }
]
```

The extension is missing the property `documentation` which is required by the declared CSAF Extension Schema.

6.1.60.3 Extension Metadata

For each element of type `#/$defs/extensions_t` it MUST be tested that the requirements provided through its metadata are fulfilled for each CSAF Extension used. CSAF Extensions not supported by the implementation SHALL result in a warning which MUST include the value of `$schema` of the extension. Such warning SHALL differentiate between the different classes of extensions.

This includes, but is not limited to:

- The extension is allowed at this path at all.
- The extension is not used more often than allowed.
- The extension is not used with other incompatible extensions.

The relevant paths for this test are:

```
/document/x_extensions
/product_tree/branches[](/branches[])*/product/x_extensions
/product_tree/full_product_names[]/x_extensions
/product_tree/product_paths[]/full_product_name/x_extensions
/vulnerabilities[]/metrics[]/content/x_extensions
/vulnerabilities[]/x_extensions
/x_extensions
```

Example 1 (which fails the test):

```
{
  // ...
  "x_extensions": [
    {
      "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/
11/documentation-11-content_1.0.0.json",
      "category": "high_value",
      // ...
    },
    {
      "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/
11/documentation-11-content_1.0.0.json",
      "category": "informational",
      // ...
    }
  ]
}
```

The extension is only allowed once in the path `/x_extensions`.

6.1.61 Use of Multiple Stars in SKU

For each stock keeping unit it MUST be tested that the it does not contain multiple unescaped stars.

Multiple `*` that match zero or multiple characters within a model number introduce ambiguity and are therefore prohibited.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/skus[]
/product_tree/full_product_names[]/product_identification_helper/skus[]
/product_tree/product_paths[]/full_product_name/product_identification_helper/skus[]
```

Example 1 (which fails the test):


```
"skus": [
  "NL*12*"
]
```

The stock keeping unit contains two unescaped stars.

6.2 Recommended Tests

Recommended tests SHOULD NOT fail at a valid CSAF document without a good reason. Failing such a test does not make the CSAF document invalid. These tests may include information about features which are still supported but expected to be deprecated in a future version of CSAF. A program MUST handle a test failure as a warning.

6.2.1 Unused Definition of Product ID

For each Product ID (type `/$defs/product_id_t`) in Full Product Name elements (type: `/$defs/full_product_name_t`) it MUST be tested that the `product_id` is referenced somewhere within the same document.

This test SHALL be skipped for CSAF documents conforming the profile “Informational Advisory”.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_id
/product_tree/full_product_names[]/product_id
/product_tree/product_paths[]/full_product_name/product_id
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
}
```

`CSAFPID-9080700` was defined but never used.

A tool MAY remove the unused definition as a quick fix. However, such quick fix shall not be applied if the test was skipped.

6.2.2 Missing Remediation

For each Product ID (type `/$defs/product_id_t`) in the Product Status groups Affected and Under investigation it MUST be tested that a remediation exists.

The remediation might be of the category `none_available` or `no_fix_planned`.

The relevant paths for this test are:


```
/vulnerabilities[]/product_status/first_affected[]
/vulnerabilities[]/product_status/known_affected[]
/vulnerabilities[]/product_status/last_affected[]
/vulnerabilities[]/product_status/under_investigation[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "product_status": {
      "last_affected": [
        "CSAFPID-9080700"
      ]
    }
  }
]
```

`CSAFPID-9080700` has in Product Status `last_affected` but there is no remediation object for this Product ID.

6.2.3 Missing Metric

For each Product ID (type `/$defs/product_id_t`) in the Product Status groups Affected it MUST be tested that a metric object exists which covers this product.

The relevant paths for this test are:

```
/vulnerabilities[]/product_status/first_affected[]
/vulnerabilities[]/product_status/known_affected[]
/vulnerabilities[]/product_status/last_affected[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
```



```
      "product_status": {
        "first_affected": [
          "CSAFPID-9080700"
        ]
      }
    }
  ]
```

`CSAFPID-9080700` has in Product Status `first_affected` but there is no metric object which covers this Product ID.

6.2.4 Build Metadata in Revision History

For each item in revision history it MUST be tested that `number` does not include build metadata.

The relevant path for this test is:

```
/document/tracking/revision_history[]/number
```

Example 1 (which fails the test):

```
"revision_history": [
  {
    "date": "2023-08-23T10:00:00.000Z",
    "number": "1.0.0+exp.sha.ac00785",
    "summary": "Initial version."
  }
]
```

The revision history contains an item which has a `number` that includes the build metadata `+exp.sha.ac00785`.

6.2.5 Older Initial Release Date than Revision History

It MUST be tested that the Initial Release Date is not older than the `date` of the oldest item in Revision History. As the timestamps might use different timezones, the sorting and comparison MUST take timezones into account.

The relevant path for this test is:

```
/document/tracking/initial_release_date
```

Example 1 (which fails the test):

```
"tracking": {
  // ...
  "initial_release_date": "2023-08-22T10:00:00.000Z",
  "revision_history": [
    {
      "date": "2023-09-06T10:00:00.000Z",
      "number": "1",
      "summary": "Initial version."
    },
```



```
      {
        "date": "2024-01-21T11:00:00.000Z",
        "number": "2",
        "summary": "Second version."
      }
    ],
    // ...
  }
```

The initial release date `2023-08-22T10:00:00.000Z` is older than `2023-09-06T10:00:00.000Z` which is the `date` of the oldest item in Revision History.

6.2.6 Older Current Release Date than Revision History

It MUST be tested that the Current Release Date is not older than the `date` of the newest item in Revision History. As the timestamps might use different timezones, the sorting and comparison MUST take timezones into account.

The relevant path for this test is:

```
/document/tracking/current_release_date
```

Example 1 (which fails the test):

```
"tracking": {
  "current_release_date": "2023-09-06T10:00:00.000Z",
  // ...
  "revision_history": [
    {
      "date": "2023-09-06T10:00:00.000Z",
      "number": "1",
      "summary": "Initial version."
    },
    {
      "date": "2024-01-21T11:00:00.000Z",
      "number": "2",
      "summary": "Second version."
    }
  ],
  // ...
}
```

The current release date `2023-09-06T10:00:00.000Z` is older than `2023-09-23T11:00:00.000Z` which is the `date` of the newest item in Revision History.

6.2.7 Missing Date in Involvements

For each item in the list of involvements it MUST be tested that it includes the property `date`.

The relevant path for this test is:

```
/vulnerabilities[]/involvements
```


Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "involvements": [
      {
        "party": "vendor",
        "status": "in_progress"
      }
    ]
  }
]
```

The list of involvements contains an item which does not contain the property `date`.

6.2.8 Use of MD5 As the Only Hash Algorithm

It MUST be tested that the hash algorithm `md5` is not the only one present.

Since collision attacks exist for MD5 such value should be accompanied by a second cryptographically stronger hash. This will allow users to double check the results.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/hashes[]
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/product_paths[]/full_product_name/product_identification_helper/hashes[
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700",
      "product_identification_helper": {
        "hashes": [
          {
            "file_hashes": [
              {
                "algorithm": "md5",
                "value": "6ae24620ea9656230f49234efd078935"
              }
            ],
            "filename": "product_a.so"
          }
        ]
      }
    }
  ]
}
```


The hash algorithm `md5` is used in one item of hashes without being accompanied by a second hash algorithm.

6.2.9 Use of SHA-1 As the Only Hash Algorithm

It MUST be tested that the hash algorithm `sha1` is not the only one present.

Since collision attacks exist for SHA-1 such value should be accompanied by a second cryptographically stronger hash. This will allow users to double check the results.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/hashes[]
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/product_paths[]/full_product_name/product_identification_helper/hashes[
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700",
      "product_identification_helper": {
        "hashes": [
          {
            "file_hashes": [
              {
                "algorithm": "sha1",
                "value": "e067035314dd8673fe1c9fc6b01414fe0950fdc4"
              }
            ],
            "filename": "product_a.so"
          }
        ]
      }
    }
  ]
}
```

The hash algorithm `sha1` is used in one item of hashes without being accompanied by a second hash algorithm.

6.2.10 Missing TLP Label (Obsolete)

The TLP label is now required by the schema. Therefore, the recommended test is obsolete. This section is kept to document that change and keep the numbering of the remaining sections stable. The test is excluded from any preset and requirement to be executed.

6.2.11 Missing Canonical URL

It MUST be tested that the CSAF document has a canonical URL.

To implement this test it is deemed sufficient that one item in `/document/references` fulfills all of the following:

- It has the category `self`.
- The `url` starts with `https://`.
- The `url` ends with the valid filename for the CSAF document according to the rules in section 5.1.

The relevant path for this test is:

```
/document/references
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "references": [
    {
      "category": "self",
      "summary": "A non-canonical URL.",
      "url": "https://example.com/security/data/csaf/2024/oasis_csaf_tc-csaf_2.1-
2024-6-2-11-01_1.json"
    }
  ],
  // ...
  "tracking": {
    // ...
    "id": "OASIS_CSAF_TC-CSAF_2.1-2024-6-2-11-01",
    // ...
    "version": "1"
  },
  // ...
}
```

The only element where the `category` is `self` has a URL that does not fulfill the requirement of a valid filename for a CSAF document.

6.2.12 Missing Document Language

It MUST be tested that the document language member is present and set. A CSAF Validator SHALL differentiate in the error message between the key being present but having no or an empty value and not being present at all.

The relevant path for this test is:

```
/document/lang
```

Example 1 (which fails the test):

```
"document": {
  "category": "csaf_base",
  "csaf_version": "2.1",
  "distribution": {
    "tlp": {
```



```
      "label": "CLEAR"
    }
  },
  "publisher": {
    // ...
  },
  // ...
}
```

The document language is not defined.

A tool MAY add the key as a quick fix. In such case, the value still needs to be set - either manually or by other means from the tool, e.g. through a given configuration.

6.2.13 Sorting

It MUST be tested that all keys in a CSAF document are sorted alphabetically.

The relevant path for this test is:

```
/
```

Example 1 (which fails the test):

```
"document": {
  "csaf_version": "2.1",
  "category": "csaf_base",
  // ...
}
```

The key `csaf_version` is not at the right place.

A tool MAY sort the keys as a quick fix.

6.2.14 Use of Private Language

For each element of type `/$defs/lang_t` it MUST be tested that the language code does not contain subtags reserved for private use.

The relevant paths for this test are:

```
/document/lang
/document/source_lang
```

Example 1 (which fails the test):

```
"lang": "qtx"
```

The language code `qtx` is reserved for private use.

A tool MAY remove such subtag as a quick fix.

6.2.15 Use of Default Language

For each element of type `/$defs/lang_t` it MUST be tested that the language code is not `i-default`.

The relevant paths for this test are:

```
/document/lang
/document/source_lang
```

Example 1 (which fails the test):

```
"lang": "i-default"
```

The language code `i-default` is used.

A tool MAY remove such element as a quick fix.

6.2.16 Missing Product Identification Helper

For each element of type `/$defs/full_product_name_t` it MUST be tested that it includes the property `product_identification_helper`.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product
/product_tree/full_product_names[]
/product_tree/product_paths[]/full_product_name
```

Example 1 (which fails the test):

```
"full_product_names": [
  {
    "product_id": "CSAFPID-9080700",
    "name": "Product A"
  }
]
```

The product `CSAFPID-9080700` does not provide any Product Identification Helper at all.

6.2.17 CVE in Field IDs

For each item in `/vulnerabilities[]/ids` it MUST be tested that it is not a CVE ID.

It is sufficient to check, whether the property `text` matches the regex `^CVE-[0-9]{4}-[0-9]{4,}$`.

The relevant paths for this test are:

```
/vulnerabilities[]/ids[]
```

Example 1 (which fails the test):

```
"ids": [
  {
    "system_name": "CVE Project",
    "text": "CVE-2021-44228"
  }
]
```

The `CVE-2021-44228` is listed in an item of the `ids` array instead under `cve`.

A tool MAY set such element as value for the `cve` property as a quick fix, if that didn't exist before. Alternatively, it MAY remove such element as a quick fix.

6.2.18 Product Version Range without vers

For each element of type `/$defs/branches_t` with `category` of `product_version_range` it MUST be tested that the value of `name` indicates that the product version range is using vers.

Compliance with the vers specification itself is enforced via test 6.1.50. Unknown version schemes are detected via test 6.2.51.
To implement this test it is deemed sufficient that the value of `name` matches the following regex:

```
^vers:[a-z\\.\\-\\+][a-z0-9\\.\\-\\+]*/.+
```

The warning MUST clearly advise to carefully check whether vers can be used or the versions can be enumerated as the use of vls is only a fallback option. For more details, see sections 3.1.2.2 and 3.1.2.3.2.

It is planned to deprecate and finally remove the support for vls in future versions of this standard.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
"branches": [
  {
    "category": "product_version_range",
    "name": ">4.2",
    // ...
  }
]
```


The version range `>4.2` is a valid vls but not valid according to the vers specification.

6.2.19 CVSS for Fixed Products

For each item the fixed products group (`first_fixed` and `fixed`) it MUST be tested that a CVSS applying to this product has an overall score of `0`. The test SHALL pass if none of the Product IDs listed within product status `fixed` or `first_fixed` is found in `products` of any item of the `metrics` element.

The relevant path for this test is:

```
/vulnerabilities[]/product_status/first_fixed[]
/vulnerabilities[]/product_status/fixed[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "product_status": {
      "fixed": [
        "CSAFPID-9080700"
      ]
    },
    "metrics": [
      {
        "content": {
          "cvss_v3": {
            "baseScore": 6.5,
            "baseSeverity": "MEDIUM",
            "vectorString": "CVSS:3.1/AV:L/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:H",
            "version": "3.1"
          }
        }          ,
        "products": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

Neither the `environmentalScore` nor the properties `modifiedAvailabilityImpact`, `modifiedConfidentialityImpact`, `modifiedIntegrityImpact` nor the corresponding attributes in the `vectorString` have been set.

A tool MAY remove any Product IDs listed within product status `fixed` or `first_fixed` from `products` of all items of the `metrics` element.
Alternatively, a tool MAY set those environmental properties according to the CVSS version used that reduce the overall score to `0` and compute the respective score and severity as a quick fix. The following environmental properties have been identified:

- CVSS v2: `targetDistribution` to `NONE`
- CVSS v3: all of `modifiedAvailabilityImpact`, `modifiedConfidentialityImpact`, and `modifiedIntegrityImpact` to `NONE`
- CVSS v4: all of
  - `modifiedVulnAvailabilityImpact`, `modifiedVulnConfidentialityImpact`, and `modifiedVulnIntegrityImpact` to `NONE` and
  - `modifiedSubAvailabilityImpact`, `modifiedSubConfidentialityImpact`, and `modifiedSubIntegrityImpact` to `NEGLIGIBLE`

6.2.20 Additional Properties

It MUST be tested that there is no additional property in the CSAF document that was not defined in the CSAF JSON schema. This also applies for referenced schemas. For CSAF Extensions, the declared schemas MUST be checked additionally to the CSAF Extension Content Schema. Additional and unevaluated properties allowed by the CSAF Extension Content Schema MUST not be reported. CSAF Extensions not supported by the implementation SHALL result in a warning which MUST include the value of `$schema` of the extension. Such warning SHALL differentiate between the different classes of extensions.

Some of the errors could be reported by combining the JSON schema check with several mandatory tests, this test provides an one stop solution and can be used as quality gate in CSAF publication pipelines or contracts. Therefore, a potential double reporting is not consider an issue.

The relevant path for this test is:

```
/
```

To implement this test it is deemed sufficient to validate the CSAF document against a “strict” version schema that has all references integrated and sets `additionalProperties` respectively `unevaluatedProperties` to `false` at all appropriate places to detect additional properties.

Example 1 (which fails the test):

```
"cvss_v3": {
  "baseScore": 6.4,
  "baseSeverity": "MEDIUM",
  "custom_property": "any",
  "vectorString": "CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:C/C:N/I:H/A:L",
  "version": "3.1"
}
```

The key `custom_property` is not defined in the JSON schema.

A tool MAY remove such keys as a quick fix.

6.2.21 Same Timestamps in Revision History

It MUST be tested that the timestamps of all items in the revision history are pairwise disjoint. As the timestamps might use different timezones, the comparison MUST take timezones into account.

The relevant path for this test is:

```
/document/tracking/revision_history[]/date
```

Example 1 (which fails the test):

```
"revision_history": [
  {
    "date": "2024-01-21T10:00:00.000Z",
    "number": "2.0.0",
    "summary": "Second version."
  },
  {
    "date": "2024-01-21T10:00:00.000Z",
    "number": "1.0.0",
    "summary": "Initial version."
  }
]
```

The first and second revision have the same timestamp.

6.2.22 Document Tracking ID in Title

It MUST be tested that the `/document/title` does not contain the `/document/tracking/id`.

The relevant path for this test is:

```
/document/title
```

Example 1 (which fails the test):

```
  "title": "OASIS_CSAF_TC-CSAF_2.1-2024-6-2-22-01: Recommended test: Document Trackir
  "tracking": {
    // ...
    "id": "OASIS_CSAF_TC-CSAF_2.1-2024-6-2-22-01",
    // ...
  }
```

The document title contains the document tracking id.

A tool MAY remove the document tracking id from the document title. It SHOULD also remove any separating characters including white space, colon, dash and brackets.

6.2.23 Usage of Deprecated CWE

For each item in the CWE array it MUST be tested that the CWE is not deprecated in the given version.

The relevant path for this test is:

```
/vulnerabilities[]/cwes[]
```

Example 1 (which fails the test):

```
"cwes": [
   {
     "id": "CWE-596",
     "name": "DEPRECATED: Incorrect Semantic Object Comparison",
     "version": "4.13"
   }
 ]
```

The `CWE-596` is deprecated in version `4.13`.

A tool MAY suggest to replace the deprecated CWE with its replacement or closest equivalent.

6.2.24 Usage of Non-Latest CWE Version

For each item in the CWE array it MUST be tested that the latest CWE version available at the time of the last revision was used. The test SHALL fail if a later CWE version was used.

The relevant path for this test is:

```
/vulnerabilities[]/cwes[]
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "tracking": {
    "current_release_date": "2024-01-21T10:00:00.000Z",
    // ...
  }
},
"vulnerabilities": [
  {
    "cwes": [
      {
        "id": "CWE-256",
        "name": "Plaintext Storage of a Password",
        "version": "4.12"
      }
    ]
  }
]
```


The CWE version listed is `4.12`. However, version `4.13` was most recent version when the document was released on `2024-01-21T10:00:00.000Z`.

A tool MAY suggest to use the latest version available at the time of the `current_release_date`. This is most likely also the overall latest CWE version as modifications to a CSAF document lead to a new `current_release_date`.

6.2.25 Usage of CWE Not Allowed for Vulnerability Mapping

For each item in the CWE array it MUST be tested that the vulnerability mapping is allowed.

Currently, this includes the two usage state `Allowed` and `Allowed-with-Review`.
Note: The property `Usage` within the `MappingNotesType` was introduced in version `7.0` of the CWE schema definition. As a consequence, this information might not be available before CWE version `4.12`.

The relevant path for this test is:

```
/vulnerabilities[]/cwes[]
```

Example 1 (which fails the test):

```
"cwes": [
  {
    "id": "CWE-20",
    "name": "Improper Input Validation",
    "version": "4.13"
  }
]
```

The usage of CWE-20 is discouraged as “is commonly misused in low-information vulnerability reports when lower-level CWEs could be used instead, or when more details about the vulnerability are available” [CWE-20].

6.2.26 Usage of CWE Allowed with Review for Vulnerability Mapping

For each item in the CWE array it MUST be tested that the vulnerability mapping is allowed without review.

Reasoning: CWEs marked with a vulnerability mapping state of `Allowed-with-Review` should only be used if a thorough review was done. This test helps to flag such mappings which can be used to trigger processes that ensure the extra review, e.g. by a senior analyst.
Note: The property `Usage` within the `MappingNotesType` was introduced in version `7.0` of the CWE schema definition. As a consequence, this information might not be available before CWE version `4.12`.

The relevant path for this test is:

```
/vulnerabilities[]/cwes[]
```

Example 1 (which fails the test):


```
"cwes": [
  {
    "id": "CWE-1023",
    "name": "Incomplete Comparison with Missing Factors",
    "version": "4.13"
  }
]
```

The usage of CWE-1023 is allowed with review as the “CWE entry is a Class and might have Base-level children that would be more appropriate”. [CWE-1023]

6.2.27 Discouraged Product Status Remediation Combination

For each item in `/vulnerabilities[]/remediations`, it MUST be tested that a Product is not member of a discouraged product status group remediation category combination (see table tab). This takes indirect relations through Product Groups into account.

The relevant path for this test is:

```
/vulnerabilities[]/remediations[]
```

Example 1 (which fails the test):

```
"product_status": {
  "known_not_affected": [
    "CSAFPID-9080700"
  ]
},
"remediations": [
  {
    "category": "fix_planned",
    "details": "The fix should be available in Q4 2024.",
    "product_ids": [
      "CSAFPID-9080700"
    ]
  }
]
```

For the product with product ID `CSAFPID-908070` a fix is planned but the product was not affected at all.

6.2.28 Usage of Max UUID

It MUST be tested that the Max UUID is not used as sharing group id.

The relevant path for this test is:

```
/document/distribution/sharing_group/id
```

Example 1 (which fails the test):


```
"distribution": {
  "sharing_group": {
    "id": "ffffffff-ffff-ffff-ffff-ffffffffffff",
    "name": "Public"
  },
  // ...
},
```

The sharing group id uses the Max UUID.

A tool MAY remove the property `sharing_group` as a quick fix.

6.2.29 Usage of Nil UUID

It MUST be tested that the Nil UUID is not used as sharing group id.

The relevant path for this test is:

```
/document/distribution/sharing_group/id
```

Example 1 (which fails the test):

```
"distribution": {
  "sharing_group": {
    "id": "ffffffff-ffff-ffff-ffff-ffffffffffff",
    "name": "Public"
  },
  // ...
},
```

The sharing group id uses the Nil UUID.

A tool MAY remove the property `sharing_group` as a quick fix.

6.2.30 Usage of Sharing Group on TLP:CLEAR

It MUST be tested that no sharing group is used if the document is `TLP:CLEAR`.

The relevant path for this test is:

```
/document/distribution/sharing_group
```

Example 1 (which fails the test):

```
"distribution": {
  "sharing_group": {
    "id": "ffffffff-ffff-ffff-ffff-ffffffffffff",
    "name": "Public"
  },
  "tlp": {
    "label": "CLEAR"
  }
},
```


The CSAF document is `TLP:CLEAR` but a sharing group is given.

A tool MAY remove the property `sharing_group` as a quick fix.

6.2.31 Hardware and Software

For each product containing at least one of the Product Identification Helpers `serial_numbers` or `model_numbers` it MUST be tested that a product path exists referencing this product.

This tests detects a potential situation where hardware and software have been mixed in the `product_tree`. Note: This test will fail if the CSAF document contains in its `product_tree` only hardware. However, this is expected and considered a good reason for the test to fail. This does not make the CSAF document invalid.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product
/product_tree/full_product_names[]
/product_tree/product_paths[]/full_product_name
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version",
              "name": "4.1",
              "product": {
                "name": "Example Company Controller A Firmware 4.1",
                "product_id": "CSAFPID-908070601",
                "product_identification_helper": {
                  "serial_numbers": [
                    "143-D-354"
                  ]
                }
              }
            }
          ],
          "category": "product_name",
          "name": "Controller A"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ]
}
```


The `product_tree` mentions the hardware product Example Company Controller A and combines it with the Firmware version 4.1.

A correct representation for hardware and software in the `product_tree` is given in example [[1 (of section 5.6)]] in section 5.6.

6.2.32 Use of Same Product Identification Helper for Different Products

For each Product Identification Helper category it MUST be tested that the same value is not used for multiple products in this category.

This test detects a potentially incorrect constructed product tree. Note: This test will fail if the CSAF document contains in its `product_tree` the old and new name of a product that was renamed. However, this is expected and considered a good reason for the test to fail. This does not make the CSAF document invalid.
For the comparison, arrays need to be treated as unordered sets which need to be pairwise disjoint. When comparing two helper-objects' array properties whose items are themselves JSON objects (e.g. `hashes` or `file_hashes`), the array needs to be treated as an unordered set, and each object is compared by deep-equality of its key/value pairs (ignoring key-order).

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper
/product_tree/full_product_names[]/product_id/product_identification_helper
/product_tree/product_paths[]/full_product_name/product_id/product_identification_hel
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version",
              "name": "1.0",
              "product": {
                "name": "Example Company Product A 1.0",
                "product_id": "CSAFPID-908070601",
                "product_identification_helper": {
                  "serial_numbers": [
                    "143-D-354"
                  ]
                }
              }
            },
            {
              "category": "product_version",
              "name": "2.0",
              "product": {
                "name": "Example Company Product A 2.0",
                "product_id": "CSAFPID-908070602",
```



```
                "product_identification_helper": {
                  "serial_numbers": [
                    "143-D-354"
                  ]
                }
              }
            }
          ],
          "category": "product_name",
          "name": "Product A"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ]
}
```

Both products are identified by the same serial number `143-D-354`.

6.2.33 Disclosure Date Newer than Revision History

For each vulnerability, it MUST be tested that the `disclosure_date` is earlier or equal to the `date` of the newest item of the `revision_history` if the `disclosure_date` is in the past at the time of the test execution. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The result of the test is dependent upon the time of the execution of the test - it might change for a given CSAF document over time. However, the latest version of a CSAF document should always pass the test.

The relevant path for this test is:

```
/vulnerabilities[]/disclosure_date
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "distribution": {
    "tlp": {
      "label": "GREEN"
    }
  },
  // ...
  "tracking": {
    "current_release_date": "2024-01-24T10:00:00.000Z",
    // ...
    "initial_release_date": "2024-01-24T10:00:00.000Z",
    "revision_history": [
      {
        "date": "2024-01-24T10:00:00.000Z",
        "number": "1",
        "summary": "Initial version."
```



```
        }
      ],
      "status": "final",
      "version": "1"
    }
  },
  "vulnerabilities": [
    {
      "disclosure_date": "2024-02-24T10:00:00.000Z"
    }
  ]
```

The `disclosure_date` is in the past but newer than the date of newest item in the `revision_history`.

6.2.34 Usage of Unknown SSVC Decision Point Base Namespace

For each SSVC decision point given under `selections`, it MUST be tested that the base `namespace` is a registered one. Namespaces reserved for special purpose MUST be treated as per their definition.

This test fails on unregistered namespaces as well as registered ones not yet supported by the implementation.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/namespace
```

Example 1 (which fails the test):

```
"ssvc_v2": {
  "schemaVersion": "2.0.0",
  "selections": [
    {
      // ...
      "namespace": "x_example.unregistered#some-yet-unknown-or-maybe-private-namespac
      // ...
    }
  ],
  // ...
}
```

The namespace `x_example.unregistered#some-yet-unknown-or-maybe-private-namespace` is not a registered namespace. Its decision point definitions might therefore not be known to the reader of the document.

6.2.35 Usage of Unregistered SSVC Decision Point Base Namespace in TLP:CLEAR Document

For each SSVC decision point given under `selections`, it MUST be tested that the base `namespace` is not an unregistered one if the document is labeled `TLP:CLEAR`. Namespaces reserved for special purpose MUST be treated as per their definition.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/namespace
```


Example 1 (which fails the test):

```
{
  "document": {
    // ...
    "distribution": {
      "tlp": {
        "label": "CLEAR"
      }
    },
    // ...
  }
  "vulnerabilities": [
    {
      "metrics": [
        {
          "content": {
            "ssvc_v2": {
              // ...
              "selections": [
                {
                  // ...
                  "namespace": "x_example.unregistered#namespace",
                  // ...
                }
              ],
              // ...
            }
          },
          // ...
        }
      ]
    }
  ]
}
```

The namespace `x_example.unregistered#namespace` is an unregistered base namespace. Its decision point definitions might therefore not be known to the reader of the document.

6.2.36 Usage of SSVC Decision Point Namespace with Extension in TLP:CLEAR Document

For each SSVC decision point given under `selections`, it MUST be tested that the `namespace` does not use an extension if the document is labeled `TLP:CLEAR`. Namespaces reserved for special purpose MUST be treated as per their definition.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/namespace
```

Example 1 (which fails the test):


```
{
  "document": {
    // ...
    "distribution": {
      "tlp": {
        "label": "CLEAR"
      }
    },
    // ...
  }
  "vulnerabilities": [
    {
      "metrics": [
        {
          "content": {
          "ssvc_v2": {
            // ...
            "selections": [
              {
                // ...
                "namespace": "ssvc//.example.test#refined-technical-impacts",
                // ...
              }
              ],
              // ...
            }
          },
          // ...
        }
      ]
    }
  ]
}
```

The namespace contains the extension `.example.test#refined-technical-impacts`. Its decision point definitions might therefore not be known to the reader of the document. However, as per SSVC specification, extensions cannot extend an existing decision point with new values. Thus, they can be treated as decision points from the base namespace, if the extension is unknown.

6.2.37 Usage of Unknown SSVC Decision Point Namespace without Resource

For each SSVC object containing a decision point with a full `namespace` that is not registered, it MUST be tested that a Decision Point Resource exists for each one that provides additional context about the decision points from this namespace. Namespaces reserved for special purpose MUST be treated as per their definition.

A full namespace includes any extension. To implement this test it is deemed sufficient to check whether the `summary` contains the full `namespace`.

Namespaces reserved for special purpose MUST be treated as per their definition.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/decision_point_resources
```


Example 1 (which fails the test):

```
  "content": {
    "ssvc_v2": {
      "decision_point_resources": [
        {
          "summary": "Specification for Decision Points within 'x_example.test#some-
other-collection'",
          "uri": "https://test.example/ssvc-collection/test-some-other-collection"
        }
      ],
      // ...
      "selections": [
        {
          // ...
          "namespace": "x_example.test#without-resource/de-DE",
          // ...
        }
      ],
      // ...
    }
  },
```

The namespace `x_example.test#without-resource/de-DE` is not a registered one, however no `decision_point_resources` mentions the `namespace` in its `summary`.

6.2.38 Usage of Deprecated Profile

It MUST be tested that the `/document/category` does not start with `csaf_deprecated_`.

The relevant path for this test is:

```
/document/category
```

Example 1 (which fails the test):

```
"category": "csaf_deprecated_security_advisory",
```

The document category starts with `csaf_deprecated_`.

6.2.39 Profile Tests

This subsubsection structures the recommended tests for the profiles. Not all tests apply for all profiles. Tests SHOULD be skipped if the document category does not match the one given in the test. Each of the following tests SHOULD be treated as they where listed similar to the other tests.

6.2.39.1 Missing Fixed Product

For each product listed in the product status group affected in any vulnerability, it MUST be tested that a corresponding version of the product is listed as fixed in the same vulnerability. The test MUST be skipped if there is a clear indication, that such a version of the product does not exist. Indicators include a remediation item with one of the categories `fix_planned`, `no_fix_planned` or `none_available` referring to the affected product. The test MUST NOT be skipped, if there is an indication, that such a version of the product might exist. Indicators include an affected product version range with the comparator < in the last version constraint and a remediation item with the categories `vendor_fix` referring to the affected product.

The relevant value for `/document/category` is:

```
csaf_security_advisory
```

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    // ...
    "product_status": {
      "known_affected": [
        "CSAFPID-9080700"
      ]
    },
    "remediations": [
      {
        "category": "vendor_fix",
        "details": "Update to the latest version, at least version 4.2.",
        "product_ids": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

The fixed product is not listed in the advisory but there is a clear indication that such product exists as there is a remediation with category `vendor_fix`.

A tool MAY create the missing fixed product based on the data available in the advisory as a quick fix.

6.2.39.2 Language Specific Reasoning for Withdrawal

If the document language is specified but not English, it MUST be tested that exactly one item in document notes exists that has the language specific translation of the term `Reasoning for Withdrawal` as `title`. The `category` of this item MUST be `description`. If no language specific translation has been recorded, the test MUST be skipped and output an information to the user that no such translation is known.

A list of the language specific translations is kept at the OASIS CSAF TC.

The relevant value for `/document/category` is:

```
csaf_withdrawn
```

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"notes": [
  {
    "category": "summary",
    "text": "Das CSAF Document enthielt Beispieldaten und wurde zurückgezogen, um T
    "title": "Begründung für die Zurückziehung"
  }
],
```

The note has the correct title. However, it uses the wrong category.

6.2.39.3 Language Specific Reasoning for Supersession

If the document language is specified but not English, it MUST be tested that exactly one item in document notes exists that has the language specific translation of the term `Reasoning for Supersession` as `title`. The `category` of this item MUST be `description`. If no language specific translation has been recorded, the test MUST be skipped and output an information to the user that no such translation is known.

A list of the language specific translations is kept at the OASIS CSAF TC.

The relevant value for `/document/category` is:

```
csaf_superseded
```

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"notes": [
  {
    "category": "summary",
    "text": "Das CSAF Dokument enthielt Beispieldaten und wurde ersetzt, um Testdat
    "title": "Begründung für die Ersetzung"
  }
],
```


> The note has the correct title. However, it uses the wrong category.

6.2.39.4 Language Specific Superseding Document

If the document language is specified but not English, it MUST be tested that at least one item in document references exists that starts with the language specific translation of the term `Superseding Document` as `summary`. The `category` of this item MUST be `external`. If no language specific translation has been recorded, the test MUST be skipped and output an information to the user that no such translation is known.

> A list of the language specific translations is kept at the OASIS CSAF TC.

The relevant value for `/document/category` is:

```
csaf_superseded
```

The relevant path for this test is:

```
/document/references
```

Example 1 (which fails the test):

```
"references": [
  {
    "category": "self",
    "summary": "Ersetztes Dokument",
    "url": "https://example.com/.well-known/csaf/clear/2024/esa-2024-1234.json"
  }
],
```

> The note has the correct title. However, it uses the wrong category.

6.2.39.4.1 Extension in Superseded or Withdrawn Document

It MUST be tested that the document does not contain an extension.

The relevant values for `/document/category` are:

```
csaf_withdrawn
csaf_superseded
```

The relevant paths for this test are:

```
/document/x_extensions
/x_extensions
```

Example 1 (which fails the test):


```
  {
    // ...
    "x_extensions": [
      {
        "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/
11/documentation-11-content_1.0.0.json",
        // ...
      }
    ]
  }
```

The document contains a CSAF Extension.

6.2.40 Product Description without Product Reference

For each product description it MUST be tested that it includes at least one of the elements `group_ids` or `product_ids`.

If the document language is English or unspecified, the product description can be identified by checking for a note containing the corresponding `category` and `title` combination from 3.2.2.8. For other languages, the language specific translation is used.

If no language specific translation has been recorded, the test MUST be skipped and output an information to the user that no translation for product description is known.

The relevant path for this test is:

```
  /document/notes[]
```

Example 1 (which fails the test):

```
    "notes": [
      {
        "category": "description",
        "text": "Product A is a local time tracking tool. It is mainly used by software
tracking systems.",
        "title": "Product Description"
      }
    ],
```

The given note item does not specify to which products it applies to.

6.2.41 Old EPSS Timestamp

For each vulnerability, it MUST be tested that the youngest EPSS `timestamp` is not more than 15 days older than to the `date` of the newest item of the `revision_history` if the document status is `final` or `interim`. As the timestamps might use different timezones, the sorting MUST take timezones into account.

The relevant path for this test is:

```
    /vulnerabilities[]/metrics[]/content/epss/timestamp
```


Example 1 (which fails the test):

```
"document": {
  // ...
  "distribution": {
    "tlp": {
      "label": "CLEAR"
    }
  },
  // ...
  "tracking": {
    // ...
    "revision_history": [
      {
        "date": "2024-01-24T10:00:00.000Z",
        "number": "1",
        "summary": "Initial version."
      }
    ],
    "status": "final",
    // ...
  }
},
"vulnerabilities": [
  {
    "cve": "CVE-1900-0001",
    "metrics": [
      {
        "content": {
          "epss": {
            "percentile": "0.999990000",
            "probability": "0.975570000",
            "timestamp": "2024-07-13T10:00:00.000Z"
          }
        },
        "products": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

The document is in status `final` but the EPSS `timestamp` is more than 15 days older than the `date` of newest item in the `revision_history`.

6.2.42 Inconsistent Product Identification Helper

For each product identification helper which resides in `branches`, it MUST be tested that the product identification helper contain at least the same information as the categorized strings. Information that cannot be represented in the specific product identification helper MUST be omitted from the comparison.

To implement this test it is deemed sufficient to follow the process below. It is based on a static mapping of branch categories to the corresponding part of the product identification helpers. The latter one are referred to as counterparts.

1. Determine counterpart in the specific product identification helper for all categories used along the path up to the product.
2. Check whether the counterparts in the product identification helper are set.
3. Check whether each set counterpart does not contain a wildcard unless the value of the categorized string would indicate that.
4. Check whether the version part algins between the categorized strings and the product identification helper.
5. For CPE only: Check whether extra information is included in the CPE that is not in the categorized string. This might be a counterpart that is set but the corresponding categorized string is missing along th path.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/cpe
/product_tree/branches[](/branches[])*/product/product_identification_helper/purl
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "category": "vendor",
      "name": "Example Company",
      "branches": [
        {
          "category": "product_name",
          "name": "Product A",
          "branches": [
            {
              "category": "product_version",
              "name": "2.2.0",
              "product": {
                "product_id": "CSAFPID-9080700",
                "name": "Example Company Product A 2.2.0",
                "product_identification_helper": {
                  "cpe": "cpe:2.3:h:example:product_a:2.2.0:*:*:de:*:*:*:*"
                }
              }
            },
            {
              "category": "product_version",
              "name": "2.3.0",
              "product": {
                "product_id": "CSAFPID-9080701",
                "name": "Example Company Product A 2.3.0",
                "product_identification_helper": {
                  "cpe": "cpe:2.3:h:example:product_a:2.3.0:PL17:*:*:*:*:*:*"
                }
              }
            }
          ]
```



```
          }
        ]
      }
    ]
  }
```

The CPE of the first product contains a language part that is not given through the categorized strings. The CPE of the second product contains an update part that is not given through the categorized strings.

6.2.43 Missing License Expression

It MUST be tested that the license expression is present and set. A CSAF Validator SHALL differentiate in the error message between the key being present but having no or an empty value and not being present at all.

The relevant path for this test is:

```
/document/license_expression
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "lang": "en-US",
  "publisher": {
    // ...
  },
  // ...
}
```

The license expression is not defined.

A tool MAY add the key as a quick fix. In such case, the value still needs to be set - either manually or by other means from the tool, e.g. through a given configuration.

6.2.44 Deprecated License Identifier

It MUST be tested that all license identifier and exceptions used are not deprecated. This SHALL be tested for the SPDX license list and AboutCode's "ScanCode LicenseDB". The test MAY be skipped for other license inventoring entities.

The relevant path for this test is:

```
/document/license_expression
```

Example 1 (which fails the test):

```
"license_expression": "GFDL-1.1",
```

The license identifier `GFDL-1.1` was deprecated as of version 3.0.

6.2.45 Non-Existing License Identifier

It MUST be tested that all license identifier and exceptions used exist. This SHALL be tested for the SPDX license list and AboutCode's "ScanCode LicenseDB". The test MAY be skipped for other license inventoring entities.

The relevant path for this test is:

```
/document/license_expression
```

Example 1 (which fails the test):

```
"license_expression": "A-License-Identifier-That-Does-Not-Exist",
```

The license identifier does not exist in the SPDX license list.

6.2.46 Language Specific License Text

If the document language is specified but not English, and the `license_expression` contains license identifiers or exceptions that are not listed in the SPDX license list or AboutCode's "ScanCode LicenseDB", it MUST be tested that exactly one item in document notes exists that has the language specific translation of the term `License` as title. The category of this item MUST be `legal_disclaimer`. If no language specific translation has been recorded, the test MUST be skipped and output an information to the user that no such translation is known.

The relevant path for this test is:

```
/document/notes
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "lang": "de-DE",
  "license_expression": "LicenseRef-www.example.com-no-work",
  "notes": [
    {
      "category": "summary",
      "text": "Es handelt sich nicht um ein urheberrechtliches Werk, da die Schöpfung
      "title": "Lizenz"
    }
  ],
  // ...
}
```

The note has the correct title. However, it uses the wrong category.

6.2.47 Use of Qualitative Severity Rating by Issuing Party

For each item in `metrics` provided by the issuing party it MUST be tested that it does not use the qualitative severity rating.

This covers all items in `metrics` that do not have a `source` property and those where the `source` is equal to the canonical URL. It does not cover assessments made by third parties.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "qualitative_severity_rating": "low"
        },
        "products": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

The document issuer provided metric for `CSAFPID-9080700` uses a `qualitative_severity_rating`.

A tool MAY recommend other metrics or guide a CVSS assessment.

6.2.48 Misuse at Vendor Name

For each item in `branches` with category `vendor` it MUST be tested that the `name` is not `Open Source`. The comparison is case and white space insensitive.

Some issuing parties use `Open Source` as a vendor name for all open source products. This usage hinders efficient matching and is most likely not true. However, if there is a vendor whose official name is exactly `Open Source` it would be expected and acceptable to fail this test. This does not apply, if the official name differs, e.g. `Open Source Ltd.`.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):


```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          // ...
          "category": "product_name",
          "name": "ISDuBA"
        }
      ],
      "category": "vendor",
      "name": "Open Source"
    }
  ]
}
```

The issuing party used `Open Source` as vendor name for the open source product `ISDuBA`.

A tool MAY replace the vendor name with the correct one as a quick fix.

6.2.49 Upper Open Ended Product Version Range

For each element of type `/$defs/branches_t` with `category` of `product_version_range`, it MUST be tested that the value of `name` is not an upper open ended product version range.

Usually, the last including version constraint of an upper open ended product version range contains as comparator either > or >=.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
{
  "category": "product_version_range",
  "name": "vers:intdot/>=4.2.0",
  // ...
}
```

The version range has no specific upper end point and states a right-open interval.

If the upper open ended product version range consists only of one version constraint, a tool MAY convert it into the product version it starts with as a quick fix, if applicable. If the upper open ended product version range consists of more than one version constraint and the upper open range is the last version constraint, a tool MAY convert it into the product version range ending with the version the upper open ended product version as a quick fix, if applicable.

6.2.50 Overlapping Product Version Range

This subsubsection structures the recommended tests for overlapping product version ranges. The following definitions apply to all tests:

- Two product version ranges `a` and `b` are overlapping if the intersection of their sets of product versions `a'` and `b'` does not result in an empty set.

  > As the intersect operation is commutative it is sufficient to test one order.

- A product version `c` overlaps with a product version range `a` if `c` is included in `a`.
- An element of type `/$defs/full_product_name_t` which has a branch category of `product_version_range` in the path leading to it, is called "element with product version range" (`EPVR`).
- The corresponding Product ID identifying such an `EPVR` is called "EPVR Product ID" (`EPVRPID`).
- For each element of type `/$defs/branches_t` with `category` of `product_version_range` which is called "currently tested product version range" (`CTPVR`), all sibling elements on the same level form the "group of sibling elements of the product version range" (`GSEPVR`).
- The group `GSEPVR` is divided in three pairwise disjoint sets:
  1. Elements that have the branch category `product_version_range` form the "product version range sibling set" (`PVRSS`).
  2. Elements that have the branch category `product_version` form the "product version sibling set" (`PVSS`).
  3. Elements that do not fall into 1 or 2 form the "other branch category sibling set" (`OBCSS`).
- `PVRSS` is divided in three pairwise disjoint sets:
  1. Elements that use valid vers form the "product version range sibling set - vers" (`PVRSS-vers`).
  2. Elements that use valid vls form the "product version range sibling set - vls" (`PVRSS-vls`).
  3. Elements that do not fall into 1 or 2 form the "product version range sibling set - invalid" (`PVRSS-invalid`).
- `PVRSS` elements can also be grouped into three pairwise disjoint sets:
  1. Elements that contain the `branches` as only optional property form the "product version range sibling set with branches" (`PVRSS+b`).
  2. Elements that contain the `product` as only optional property form the "product version range sibling set with leaf" (`PVRSS+l`).
  3. Elements that do not fall into 1 or 2 form the "product version range sibling set - invalid branch" (`PVRSS+i`).
- Similar, `PVSS` elements can be grouped into three pairwise disjoint sets:
  1. Elements that contain the `branches` as only optional property form the "product version sibling set with branches" (`PVSS+b`).
  2. Elements that contain the `product` as only optional property form the "product version sibling set with leaf" (`PVSS+l`).
  3. Elements that do not fall into 1 or 2 form the "product version sibling set - invalid branch" (`PVSS+i`).
- Attributes of `PVRSS` elements can be combined:
  1. Product version range sibling set with branches - vers (`PVRSS+b-vers`): Intersection of `PVRSS+b` and `PVRSS-vers`
  2. Product version range sibling set with branches - vls (`PVRSS+b-vls`): Intersection of `PVRSS+b` and `PVRSS-vls`
  3. Product version range sibling set with leaf - vers (`PVRSS+l-vers`): Intersection of `PVRSS+l` and `PVRSS-vers`
  4. Product version range sibling set with leaf - vls (`PVRSS+l-vls`): Intersection of `PVRSS+l` and `PVRSS-vls`

5. Product version range sibling set with branches - invalid (`PVRSS+b-invalid`): Intersection of `PVRSS+b` and `PVRSS-invalid`
6. Product version range sibling set with leaf - invalid (`PVRSS+l-invalid`): Intersection of `PVRSS+l` and `PVRSS-invalid`
7. Product version range sibling set - invalid branch - vers (`PVRSS+i-vers`): Intersection of `PVRSS+i` and `PVRSS-vers`
8. Product version range sibling set - invalid branch - vls (`PVRSS+i-vls`): Intersection of `PVRSS+i` and `PVRSS-vls`
9. Product version range sibling set - invalid branch - invalid (`PVRSS+i-invalid`): Intersection of `PVRSS+i` and `PVRSS-invalid`

Elements in the sets `PVRSS+i` and `PVSS+i` are detectable via the validation of the JSON schema and therefore ignored in these tests. The exact characteristic of the product version range is of minor importance as the tests cannot produce any meaningful result in such an invalid product tree. Elements in `PVRSS-invalid` should be detected by test 6.1.50.

6.2.50.1 Overlapping Product Version Range with vers in Contradicting Product Status Group

For each item in `/vulnerabilities` all `EPVRPID` in the product status groups MUST be identified. For each `EPVR` (as `CTPVR`), it MUST be tested that the Product IDs of all elements in `PVRSS+l-vers` that overlap with `CTPVR` are not member of a contradicting product status groups (see section 3.2.4.12).

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version_range",
              "name": "vers:intdot/>2.1.0|<4.2.0",
              "product": {
                "name": "Example Company Product A >2.1.0|<4.2.0",
                "product_id": "CSAFPID-9080700"
              }
            },
            {
              "category": "product_version_range",
              "name": "vers:intdot/>1.2.0|<2.4.0",
              "product": {
                "name": "Example Company Product A >1.2.0|<2.4.0",
                "product_id": "CSAFPID-9080701"
              }
            }
          ],
          "category": "product_name",
          "name": "Product A"
```



```
          }
        ],
        "category": "vendor",
        "name": "Example Company"
      }
    ]
  },
  "vulnerabilities": [
    {
      "product_status": {
        "known_affected": [
          "CSAFPID-9080700"
        ],
        "known_not_affected": [
          "CSAFPID-9080701"
        ]
      }
    }
  ]
```

The product version ranges of `CSAFPID-9080700` and `CSAFPID-9080701` overlap but they are members of the contradicting groups "Affected" and "Not affected". For the version range `vers:intdot/>2.1.0|<2.4.0` it cannot be determined which product status is applicable.

A tool MAY split overlapping product version ranges into non-overlapping ones to simplify the resolution as a quick fix.

6.2.50.2 Overlapping Product Version Range with vls in Contradicting Product Status Group

For each item in `/vulnerabilities` all `EPVRPID` in the product status groups MUST be identified. For each `EPVR` (as `CTPVR`), it MUST be tested that the Product IDs of all elements in `PVRSS+l-vls` that overlap with `CTPVR` are not member of a contradicting product status groups (see section 3.2.4.12).

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version_range",
              "name": ">2.1.0|<4.2.0",
              "product": {
                "name": "Example Company Product A >2.1.0|<4.2.0",
                "product_id": "CSAFPID-9080700"
              }
```



```
            },
            {
              "category": "product_version_range",
              "name": ">1.2.0|<2.4.0",
              "product": {
                "name": "Example Company Product A >1.2.0|<2.4.0",
                "product_id": "CSAFPID-9080701"
              }
            }
          ],
          "category": "product_name",
          "name": "Product A"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ]
},
"vulnerabilities": [
  {
    "product_status": {
      "known_affected": [
        "CSAFPID-9080700"
      ],
      "known_not_affected": [
        "CSAFPID-9080701"
      ]
    }
  }
]
```

The product version ranges of `CSAFPID-9080700` and `CSAFPID-9080701` overlap but they are members of the contradicting groups "Affected" and "Not affected". For the version range `>2.1.0|<2.4.0` it cannot be determined which product status is applicable.

A tool MAY split overlapping product version ranges into non-overlapping ones to simplify the resolution as a quick fix.

6.2.50.3 Overlapping Product Version Range with Product Version in Contradicting Product Status Group

For each item in `/vulnerabilities` all `EPVRPID` in the product status groups MUST be identified. For each `EPVR` (as `CTPVR`), it MUST be tested that the Product IDs of all elements in `PVSS+l` that overlap with `CTPVR` are not member of a contradicting product status groups (see section 3.2.4.12).

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):


```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version_range",
              "name": "vers:intdot/>2.1.0|<4.2.0",
              "product": {
                "name": "Example Company Product A >2.1.0|<4.2.0",
                "product_id": "CSAFPID-9080700"
              }
            },
            {
              "category": "product_version",
              "name": "2.4.0",
              "product": {
                "name": "Example Company Product A 2.4.0",
                "product_id": "CSAFPID-9080701"
              }
            }
          ],
          "category": "product_name",
          "name": "Product A"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ]
},
"vulnerabilities": [
  {
    "product_status": {
      "known_affected": [
        "CSAFPID-9080700"
      ],
      "known_not_affected": [
        "CSAFPID-9080701"
      ]
    }
  }
]
```

The product version ranges of `CSAFPID-9080700` and `CSAFPID-9080701` overlap but they are members of the contradicting groups “Affected” and “Not affected”. For the product version `2.4.0` it cannot be determined which product status is applicable.

A tool MAY exclude the overlapping product version from the product version range as a quick fix.

6.2.51 Unknown Version Scheme in vers

For each element of type `/$defs/branches_t` with `category` of `product_version_range` which indicates that it is using vers, it MUST be tested that the version scheme is supported by the implementation. The warning MUST differentiate between officially registered version schemes and those that are not in this state.

> Different implementations might support different version schemes. Usually, unknown version schemes hinder the automated evaluation of vers. However, it is expected that the test is able to recognize all officially registered version schemes. The differentiation will help users analyzing the result of the test and addressing the issue appropriately.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
{
  "category": "product_version_range",
  "name": "vers:someweirdunknownversionscheme/<4.2.0|>3.91.1|!=3.2.0|<1.2",
  // ...
}
```

> The version scheme `someweirdunknownversionscheme` is a not an officially registered vers version scheme. Note: An implementation would also have to state whether it supports this version scheme.

6.2.52 Unknown Hash Algorithm

For each element of type `/$defs/full_product_name_t/product_identification_helper/hashes[]/file_` it MUST be tested that the hash algorithm is supported by the implementation. The warning MUST differentiate between the values mentioned in section 3.1.4.3.2 and those not mentioned there.

> Different implementations might support different hash algorithms. Usually, unknown hash algorithms might hinder the automated matching of hashes. However, it is expected that the test is able to recognize all algorithms mentioned in the standard. The differentiation will help users analyzing the result of the test and addressing the issue appropriately.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/hashes[]
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/product_paths[]/full_product_name/product_identification_helper/hashes[
```

Example 1 (which fails the test):

```
{
  "algorithm": "unknown-algorithm",
  "value": "026a37919b182ef7c63791e82c9645e2f897a3f0b73c7a6028c7febf62e93838"
}
```

> The hash algorithm `unknown-algorithm` is not listed in section 3.1.4.3.2. Note: An implementation would also have to state whether it supports this algorithm.

6.2.53 Matching Text for Registered ID System

For each item in `/vulnerabilities[]/ids` that the value of a registered vulnerability ID system as `system_name`, it MUST be tested that the `text` in the CSAF document matches the `text_pattern` given by the "Registry for Vulnerability ID Systems for CSAF" (RVISC).

> The RVISC is available at [RVISC].

The relevant path for this test is:

```
/vulnerabilities[]/ids[]/text
```

Example 1 (which fails the test):

```
"ids": [
  {
    "system_name": "https://github.com/oasis-tcs/csaf",
    "text": "Issue 1217"
  }
]
```

> The `system_name` contains a registered vulnerability ID system but the value of `text` does not match the `text_pattern` stated for "OASIS Open CSAF TC GitHub Issues" in the RVISC.

6.2.54 Extension Tests

This subsubsection structures the recommended tests for extensions. Each of the following tests SHOULD be treated as they where listed similar to the other tests.

> An application MAY group these tests when providing the additional function to only run one or more selected tests.

6.2.54.1 Registered Extension

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that the item is a registered CSAF Extension. The test MUST be skipped for official CSAF Extensions.

The relevant paths for this test are:

```
/document/x_extensions[]
/product_tree/branches[](/branches[])*/product/x_extensions[]
/product_tree/full_product_names[]/x_extensions[]
/product_tree/product_paths[]/full_product_name/x_extensions[]
/vulnerabilities[]/metrics[]/content/x_extensions[]
/vulnerabilities[]/x_extensions[]
/x_extensions[]
```

Example 1 (which fails the test):


```
  "x_extensions": [
    {
      "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/cs
11/documentation-11-content_1.0.0.json",
      // ...
    }
  ]
```

The extension is neither an official nor a registered CSAF Extension.

6.2.54.2 Official Extension

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that the item is an official CSAF Extension.

The relevant paths for this test are:

```
/document/x_extensions[]
/product_tree/branches[](/branches[])*/product/x_extensions[]
/product_tree/full_product_names[]/x_extensions[]
/product_tree/product_paths[]/full_product_name/x_extensions[]
/vulnerabilities[]/metrics[]/content/x_extensions[]
/vulnerabilities[]/x_extensions[]
/x_extensions[]
```

Example 1 (which fails the test):

```
  "x_extensions": [
    {
      "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/cs
11/documentation-11-content_1.0.0.json",
      // ...
    }
  ]
```

The extension is not an official CSAF Extension.

6.2.54.3 Critical Extension

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that the item is not a critical extension.

It is sufficient to check whether the value of `critical` is `false`.

The relevant paths for this test are:

```
/document/x_extensions[]/critical
/product_tree/branches[](/branches[])*/product/x_extensions[]/critical
/product_tree/full_product_names[]/x_extensions[]/critical
/product_tree/product_paths[]/full_product_name/x_extensions[]/critical
/vulnerabilities[]/metrics[]/content/x_extensions[]/critical
/vulnerabilities[]/x_extensions[]/critical
/x_extensions[]/critical
```


Example 1 (which fails the test):

```
"x_extensions": [
  {
    // ...
    "critical": true
  }
]
```

The extension is critical.

6.2.54.4 Usage of Experimental Extension in TLP:CLEAR Document

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that no experimental extension is used if the document is labeled `TLP:CLEAR`.

The relevant paths for this test are:

```
/document/x_extensions[]
/product_tree/branches[](/branches[])*/product/x_extensions[]
/product_tree/full_product_names[]/x_extensions[]
/product_tree/product_paths[]/full_product_name/x_extensions[]
/vulnerabilities[]/metrics[]/content/x_extensions[]
/vulnerabilities[]/x_extensions[]
/x_extensions[]
```

Example 1 (which fails the test):

```
{
  // ...
  "document": {
    // ...
    "distribution": {
      "tlp": {
        "label": "CLEAR"
      }
    },
    // ...
  }
  "x_extensions": [
    {
      "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/
11/documentation-11-content_1.0.0.json",
      // ...
    }
  ]
}
```

The extension is an experimental CSAF Extension. Its properties and meaning might therefore not be known to the reader of the document.

6.3 Informative Tests

Informative tests provide insights in common mistakes and bad practices. They MAY fail at a valid CSAF document. It is up to the issuing party to decide whether this was an intended behavior and can be ignore or should be treated. These tests MAY include information about recommended usage. A program MUST handle a test failure as a information.

6.3.1 Use of CVSS v2 As the Only Scoring System

For each item in the list of metrics which contains the `cvss_v2` object under `content` it MUST be tested that is not the only scoring item present. The test SHALL pass if a second scoring object is available regarding the specific product.

> One source might just provide CVSS v2. As long as at least one different source provides a different scoring system for the same products, the test passes.

The relevant path for this test is:

```
/vulnerabilities[]/metrics
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "cvss_v2": {
            "version": "2.0",
            "vectorString": "AV:N/AC:L/Au:N/C:C/I:C/A:C",
            "baseScore": 10
          }
        },
        "products": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

> There is only a CVSS v2 score given for `CSAFPID-9080700`.

Recommendation:

It is recommended to (also) use the CVSS v4.0.

6.3.2 Use of CVSS v3.0

For each item in the list of metrics which contains the `cvss_v3` object under `content` it MUST be tested that CVSS v3.0 is not used.

The relevant paths for this test are:

```
/vulnerabilities[]/metrics[]/content/cvss_v3/version
/vulnerabilities[]/metrics[]/content/cvss_v3/vectorString
```

Example 1 (which fails the test):

```
"cvss_v3": {
  "version": "3.0",
  "vectorString": "CVSS:3.0/AV:L/AC:L/PR:H/UI:R/S:U/C:H/I:H/A:H",
  "baseScore": 6.5,
  "baseSeverity": "MEDIUM"
}
```

The CVSS v3.0 is used.

Recommendation:

It is recommended to upgrade to CVSS v3.1.

A tool MAY upgrade to CVSS v3.1 as a quick fix. However, if such quick fix is supported the tool SHALL also recompute the `baseScore` and `baseSeverity`. The same applies for `temporalScore` and `temporalSeverity` respectively `environmentalScore` and `environmentalSeverity` if the necessary fields for computing their value are present and set.

6.3.3 Missing CVE

It MUST be tested that the CVE number is given.

The relevant path for this test is:

```
/vulnerabilities[]/cve
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "title": "BlueKeep"
  }
]
```

The CVE number is not given.

Recommendation:

It is recommended to provide a CVE number to support the users efforts to find more details about a vulnerability and potentially track it through multiple advisories. If no CVE exists for that vulnerability, it is recommended to get one assigned.

6.3.4 Missing CWE

It MUST be tested that at least one CWE is given.

The relevant path for this test is:

```
/vulnerabilities[]/cwes
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "cve": "CVE-2019-0708",
    "title": "BlueKeep"
  }
]
```

No CWE number is given.

6.3.5 Use of Short Hash

It MUST be tested that the length of the hash value is not shorter than 64 characters.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/hashes[]
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/product_paths[]/full_product_name/product_identification_helper/hashes[
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700",
      "product_identification_helper": {
        "hashes": [
          {
            "file_hashes": [
              {
                "algorithm": "md4",
                "value": "3202b50e2e5b2fcd75e284c3d9d5f8d6"
              }
            ],
            "filename": "product_a.so"
          }
        ]
      }
    }
  ]
}
```


The length of the hash value is only 32 characters long.

6.3.6 Use of Non-self Referencing URLs Failing to Resolve

For each URL which is not in the category `self` it MUST be tested that it resolves with a HTTP status code from the 2xx (Successful) or 3xx (Redirection) class.

This test does not apply for any item in an array of type `references_t` with the category `self`. For details about the HTTP status code classes see [RFC7231].

The relevant paths for this test are:

```
/document/acknowledgments[]/urls[]
/document/aggregate_severity/namespace
/document/distribution/tlp/url
/document/references[]/url
/document/publisher/namespace
/product_tree/branches[]/product/product_identification_helper/sbom_urls[]
/product_tree/branches[]/product/product_identification_helper/x_generic_uris[]/names
/product_tree/branches[]/product/product_identification_helper/x_generic_uris[]/uri
/product_tree/branches[](/branches[])*/product/product_identification_helper/sbom_url
/product_tree/branches[](/branches[])*/product/product_identification_helper/x_generi
/product_tree/branches[](/branches[])*/product/product_identification_helper/x_generi
/product_tree/full_product_names[]/product_identification_helper/sbom_urls[]
/product_tree/full_product_names[]/product_identification_helper/x_generic_uris[]/nam
/product_tree/full_product_names[]/product_identification_helper/x_generic_uris[]/uri
/product_tree/product_paths[]/full_product_name/product_identification_helper/sbom_ur
/product_tree/product_paths[]/full_product_name/product_identification_helper/x_gener
/product_tree/product_paths[]/full_product_name/product_identification_helper/x_gener
/vulnerabilities[]/acknowledgments[]/urls[]
/vulnerabilities[]/references[]/url
/vulnerabilities[]/remediations[]/url
```

Example 1 (which fails the test):

```
"references": [
  {
    "summary": "A URL that does not resolve with HTTP status code in the interval b
    "url": "https://example.invalid"
  }
]
```

The `category` is not set and therefore treated as its default value `external`. A request to that URL does not resolve with a status code from the 2xx (Successful) or 3xx (Redirection) class.

6.3.7 Use of Self Referencing URLs Failing to Resolve

For each item in an array of type `references_t` with the category `self` it MUST be tested that the URL referenced resolves with a HTTP status code less than 400.

This test will most likely fail if the CSAF document is in a status before the initial release. For details about the HTTP status code classes see [RFC7231].

The relevant paths for this test are:

```
/document/references[]/url
/vulnerabilities[]/references[]/url
```

Example 1 (which fails the test):

```
"references": [
  {
    "category": "self",
    "summary": "A URL that does not resolve with HTTP status code in the interval b
    "url": "https://example.invalid"
  }
]
```

The `category` is `self` and a request to that URL does not resolve with a status code from the 2xx (Successful) or 3xx (Redirection) class.

6.3.8 Spell Check

If the document language is given it MUST be tested that a spell check for the given language does not find any mistakes. The test SHALL be skipped if the document language is not set. It SHALL fail if the given language is not supported. The value of `/document/category` SHOULD NOT be tested if the CSAF document does not use the profile “CSAF Base”.

The relevant paths for this test are:

```
/document/acknowledgments[]/names[]
/document/acknowledgments[]/organization
/document/acknowledgments[]/summary
/document/aggregate_severity/text
/document/category
/document/distribution/text
/document/notes[]/audience
/document/notes[]/text
/document/notes[]/title
/document/publisher/issuing_authority
/document/publisher/name
/document/references[]/summary
/document/title
/document/tracking/aliases[]
/document/tracking/generator/engine/name
/document/tracking/revision_history[]/summary
/product_tree/branches[](/branches[])*/name
/product_tree/branches[](/branches[])*/product/name
/product_tree/branches[]/name
/product_tree/branches[]/product/name
/product_tree/full_product_names[]/name
/product_tree/product_groups[]/summary
```



```
/product_tree/product_paths[]/full_product_name/name
/vulnerabilities[]/acknowledgments[]/names[]
/vulnerabilities[]/acknowledgments[]/organization
/vulnerabilities[]/acknowledgments[]/summary
/vulnerabilities[]/involvements[]/summary
/vulnerabilities[]/notes[]/audience
/vulnerabilities[]/notes[]/text
/vulnerabilities[]/notes[]/title
/vulnerabilities[]/references[]/summary
/vulnerabilities[]/remediations[]/details
/vulnerabilities[]/remediations[]/entitlements[]
/vulnerabilities[]/remediations[]/restart_required/details
/vulnerabilities[]/threats[]/details
/vulnerabilities[]/title
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "lang": "en",
  "notes": [
    {
      "category": "summary",
      "text": "Secruity researchers found multiple vulnerabilities in XYZ."
    }
  ],
  // ...
}
```

There is a spelling mistake in `Secruity`.

6.3.9 Branch Categories

For each element of type `/$defs/full_product_name_t` in `/product_tree/branches` it MUST be tested that ancestor nodes along the path exist which use the following branch categories `vendor` -> `product_name` -> `product_version` in that order starting with the Product tree node.

Other branch categories can be used before, after or between the aforementioned branch categories without making the test invalid.

The relevant paths for this test are:

```
/product_tree/branches
```

Example 1 (which fails the test):

```
  "branches": [
    {
      "category": "vendor",
      "name": "Example Company",
      "branches": [
```



```
        {
          "category": "product_name",
          "name": "Product A",
          "branches": [
            {
              "category": "patch_level",
              "name": "91",
              "product": {
                "product_id": "CSAFPID-0002",
                "name": "Example Company Product A Update 91"
              }
            }
          ]
        }
      ]
    }
  ]
```

The product `CSAFPID-9080700` does not have any ancestor with the branch category `product_version`.

6.3.10 Usage of Product Version Range

For each element of type `/$defs/branches_t` it MUST be tested that the `category` is not `product_version_range`.

It is usually hard decide for machines whether a product version matches a product version ranges. Therefore, it is recommended to avoid version ranges and enumerate versions wherever possible.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/category
```

Example 1 (which fails the test):

```
"category": "product_version_range",
```

The category `product_version_range` was used.

6.3.11 Usage of V as Version Indicator

For each element of type `/$defs/branches_t` with `category` of `product_version` it MUST be tested that the value of `name` does not start with `v` or `V` before the version.

To implement this test it is deemed sufficient that the value of `name` does not match the following regex:

```
^[vV][0-9].*$
```

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):


```
          "branches": [
            {
              "category": "product_version",
              "name": "v4.2",
              // ...
            }
          ]
```

The product version starts with a `v`.

6.3.12 Missing CVSS v4.0

For each item in the list of metrics that contains any CVSS object it MUST be tested that a `cvss_v4` object is present. The test MUST fail, if any Product ID (type `/$defs/product_id_t`) in the product status group Affected (see section 3.2.4.12) is not covered by any CVSS object.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "cvss_v3": {
            "version": "3.1",
            "vectorString": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H",
            "baseScore": 10,
            "baseSeverity": "CRITICAL"
          }
        },
        "products": [
          "CSAFPID-9080700"
        ]
      }
    ]
  }
]
```

There is no CVSS v4.0 score given for `CSAFPID-9080700`.

6.3.13 Usage of Non-Latest SSVC Decision Point Version

For each SSVC decision point given under `selections` with a registered `namespace`, it MUST be tested the latest decision point `version` available at the time of the `timestamp` was used. The test SHALL fail if a later `version` was used. Namespaces reserved for special purpose MUST be treated as per their definition.

> A list of all valid decision points of registered namespaces including their values is available at the SSVC repository (see [SSVC-DP]).

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]
```

Example 1 (which fails the test):

```
"ssvc_v2": {
  // ...
  "selections": [
    {
      "key": "MI",
      "name": "Mission Impact",
      "namespace": "ssvc",
      // ...
      "version": "1.0.0"
    }
  ],
  "timestamp": "2024-01-24T10:00:00.000Z"
}
```

> At the timestamp `2024-01-24T10:00:00.000Z` version `2.0.0` of the SSVC decision point `Mission Impact` was already available.

6.3.14 Usage of Unregistered SSVC Decision Point Base Namespace in Non TLP:CLEAR Document

For each SSVC decision point given under `selections`, it MUST be tested that the base `namespace` is not an unregistered one if the document is not labeled `TLP:CLEAR`. Namespaces reserved for special purpose MUST be treated as per their definition.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/namespace
```

Example 1 (which fails the test):

```
{
  "document": {
    // ...
    "distribution": {
      "tlp": {
        "label": "GREEN"
      }
    },
```



```
      // ...
    }
    "vulnerabilities": [
      {
        "metrics": [
          {
            "content": {
              "ssvc_v2": {
                // ...
                "selections": [
                  {
                    // ...
                    "namespace": "x_example.unregistered#namespace",
                    // ...
                  }
                ],
                // ...
              }
            },
            // ...
          }
        ]
      }
    ]
  }
```

The namespace `x_example.unregistered#namespace` is an unregistered base namespace. Its decision point definitions might therefore not be known to the reader of the document.

6.3.15 Usage of SSVC Decision Point Namespace with Extension in Non TLP:CLEAR Document

For each SSVC decision point given under `selections`, it MUST be tested that the `namespace` does not use an extension if the document is not labeled `TLP:CLEAR`. Namespaces reserved for special purpose MUST be treated as per their definition.

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/namespace
```

Example 1 (which fails the test):

```
{
  "document": {
    // ...
    "distribution": {
      "tlp": {
        "label": "GREEN"
      }
    },
    // ...
  }
  "vulnerabilities": [
```



```
    {
      "metrics": [
        {
          "content": {
            "ssvc_v2": {
              // ...
              "selections": [
                {
                  // ...
                  "namespace": "ssvc//.example.test#refined-technical-impacts",
                  // ...
                }
              ],
              // ...
            }
          },
          // ...
        }
      ]
    }
  ]
}
```

The namespace contains the extension `.example.test#refined-technical-impacts`. Its decision point definitions might therefore not be known to the reader of the document.

6.3.16 Grammar Check

If the document language is given it MUST be tested that a grammar check for the given language does not find any mistakes. The test SHALL be skipped if the document language is not set. It SHALL fail if the given language is not supported.

The relevant paths for this test are:

```
/document/acknowledgments[]/summary
/document/aggregate_severity/text
/document/distribution/text
/document/notes[]/audience
/document/notes[]/text
/document/notes[]/title
/document/publisher/issuing_authority
/document/references[]/summary
/document/title
/document/tracking/revision_history[]/summary
/product_tree/product_groups[]/summary
/vulnerabilities[]/acknowledgments[]/summary
/vulnerabilities[]/involvements[]/summary
/vulnerabilities[]/notes[]/audience
/vulnerabilities[]/notes[]/text
/vulnerabilities[]/notes[]/title
/vulnerabilities[]/references[]/summary
/vulnerabilities[]/remediations[]/details
/vulnerabilities[]/remediations[]/entitlements[]
```



```
/vulnerabilities[]/remediations[]/restart_required/details
/vulnerabilities[]/threats[]/details
/vulnerabilities[]/title
```

Example 1 (which fails the test):

```
"document": {
  // ...
  "lang": "en",
  "notes": [
    {
      "category": "summary",
      "text": "The security hardening guide must followed for ensure secure operation
    }
  ],
  // ...
}
```

Multiple grammar mistakes exist:

- a `be` is missing between `must` and `followed`
- the `for` needs to be a `to`
- `a products` is also incorrect as `a` indicates singular.

6.3.17 Use of Unregistered License

It MUST be tested that the all license identifiers and exceptions are listed either in the official SPDX license identifier list or AboutCode's "ScanCode LicenseDB".

The relevant path for this test is:

```
/document/license_expression
```

Example 1 (which fails the test):

```
"license_expression": "LicenseRef-www.example.com-no-work-pd",
```

The `license_expression` contains a license identifier that is neither listed in the official SPDX license identifier list nor AboutCode's "ScanCode LicenseDB".

6.3.18 Use of Qualitative Severity Rating

For each item in `metrics` it MUST be tested that it does not use the qualitative severity rating.

This covers all items in `metrics` regardless of their origin. Even though the Qualitative Severity Rating is a specified property, it's usage is discouraged as it provides no information about the assessment inputs. Nevertheless, it can be useful to provide the such limited assessment result, especially for supply chain vulnerabilities where the upstream product just provides a qualitative severity rating.

The relevant path for this test is:


```
/vulnerabilities[]/metrics[]
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "qualitative_severity_rating": "low"
        },
        "products": [
          "CSAFPID-9080700"
        ],
        "source": "https://upstream.example/advisories/42/an-advisory-with-just-a-
severity-without-other-metrics"
      }
    ]
  }
]
```

The upstream provided metric for `CSAFPID-9080700` uses a `qualitative_severity_rating`.

6.3.19 Overlapping Product Version Range

This subsubsection structures the informative tests for overlapping product version ranges. These tests provide weak indicators of potential errors in the construction of the product tree. The abbreviations for groups are defined in section 6.2.50.

6.3.19.1 Overlapping Product Version Range with vers in Same Product Status Group

For each item in `/vulnerabilities` all `EPVRPID` in the product status groups MUST be identified. For each `EPVR` (as `CTPVR`), it MUST be tested that the Product IDs of all elements in `PVRSS+l-vers` that overlap with `CTPVR` are not member of the same product status group (see section 3.2.4.12).

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):


```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version_range",
              "name": "vers:intdot/>2.1.0|<4.2.0",
              "product": {
                "name": "Example Company Product A >2.1.0|<4.2.0",
                "product_id": "CSAFPID-9080700"
              }
            },
            {
              "category": "product_version_range",
              "name": "vers:intdot/>1.2.0|<2.4.0",
              "product": {
                "name": "Example Company Product A >1.2.0|<2.4.0",
                "product_id": "CSAFPID-9080701"
              }
            }
          ],
          "category": "product_name",
          "name": "Product A"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ]
},
"vulnerabilities": [
  {
    "product_status": {
      "first_fixed": [
        "CSAFPID-9080701"
      ],
      "fixed": [
        "CSAFPID-9080700"
      ]
    }
  }
]
```

The product version ranges of `CSAFPID-9080700` and `CSAFPID-9080701` overlap and they are members of the same group “Fixed”.

A tool MAY split overlapping product version ranges into non-overlapping ones to simplify the resolution as a quick fix.

6.3.19.2 Overlapping Product Version Range with vls in Same Product Status Group

For each item in `/vulnerabilities` all `EPVRPID` in the product status groups MUST be identified. For each `EPVR` (as `CTPVR`), it MUST be tested that the Product IDs of all elements in `PVRSS+l-vls` that overlap with `CTPVR` are not member of the same product status group (see section 3.2.4.12).

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version_range",
              "name": ">2.1.0|<4.2.0",
              "product": {
                "name": "Example Company Product A >2.1.0|<4.2.0",
                "product_id": "CSAFPID-9080700"
              }
            },
            {
              "category": "product_version_range",
              "name": ">1.2.0|<2.4.0",
              "product": {
                "name": "Example Company Product A >1.2.0|<2.4.0",
                "product_id": "CSAFPID-9080701"
              }
            }
          ],
          "category": "product_name",
          "name": "Product A"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ]
},
"vulnerabilities": [
  {
    "product_status": {
      "first_fixed": [
        "CSAFPID-9080701"
      ],
      "fixed": [
        "CSAFPID-9080700"
      ]
```



```
      }
    }
  ]
```

The product version ranges of CSAFPID-9080700 and CSAFPID-9080701 overlap and they are members of the same group “Fixed”.

A tool MAY split overlapping product version ranges into non-overlapping ones to simplify the resolution as a quick fix.

6.3.19.3 Overlapping Product Version Range with Product Version in Same Product Status Group

For each item in /vulnerabilities all EPVRPID in the product status groups MUST be identified. For each EPVR (as CTPVR), it MUST be tested that the Product IDs of all elements in PVSS that overlap with CTPVR are not member of the same product status group (see section 3.2.4.12).

The relevant path for this test is:

```
/vulnerabilities[]/product_status
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version_range",
              "name": "vers:intdot/>2.1.0|<4.2.0",
              "product": {
                "name": "Example Company Product A >2.1.0|<4.2.0",
                "product_id": "CSAFPID-9080700"
              }
            },
            {
              "category": "product_version",
              "name": "2.4.0",
              "product": {
                "name": "Example Company Product A 2.4.0",
                "product_id": "CSAFPID-9080701"
              }
            }
          ],
          "category": "product_name",
          "name": "Product A"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
```



```
    ]
  },
  "vulnerabilities": [
    {
      "product_status": {
        "first_fixed": [
          "CSAFPID-9080701"
        ],
        "fixed": [
          "CSAFPID-9080700"
        ]
      }
    }
  ]
```

The product version ranges of `CSAFPID-9080700` and `CSAFPID-9080701` overlap and they are members of the same group "Fixed".

A tool MAY exclude the overlapping product version from the product version range as a quick fix.

6.3.19.4 Overlapping Product Version Range with Product Version Range in Branch

For each item in `/vulnerabilities` all `EPVRPID` in the product status groups MUST be identified. For each `EPVR` (as `CTPVR`), it MUST be tested that all product version ranges of elements in `PVRSS+b` do not overlap with `CTPVR`.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
{
  "branches": [
    {
      "branches": [
        // ...
      ],
      "category": "product_version_range",
      "name": "vers:intdot/>2.1.0|<4.2.0"
    },
    {
      "branches": [
        // ...
      ],
      "category": "product_version_range",
      "name": "vers:intdot/>1.2.0|<2.4.0"
    }
  ],
  // ...
}
```

The product version ranges overlap.

A tool MAY split overlapping product version ranges into non-overlapping ones to simplify the resolution as a quick fix.

6.3.19.5 Overlapping Product Version Range with Product Version in Branch

For each item in `/vulnerabilities` all `EPVRPID` in the product status groups MUST be identified. For each `EPVR` (as `CTPVR`), it MUST be tested that all product versions of elements in `PVSS+b` do not overlap with `CTPVR`.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/name
```

Example 1 (which fails the test):

```
{
  "branches": [
    {
      "branches": [
        // ...
      ],
      "category": "product_version_range",
      "name": "vers:intdot/>2.1.0|<4.2.0"
    },
    {
      "branches": [
        // ...
      ],
      "category": "product_version",
      "name": "3.0.0"
    }
  ],
  // ...
}
```

The product version is part of the product version range.

6.3.20 Use of Unregistered ID System

For each item in `/vulnerabilities[]/ids` it MUST be tested that the the value of `system_name` belongs to a registered vulnerability ID system in RVISC.

The RVISC is available at [RVISC].

The relevant paths for this test are:

```
/vulnerabilities[]/ids[]/system_name
```

Example 1 (which fails the test):


```
"ids": [
  {
    "system_name": "OASIS Open CSAF TC GitHub Issues",
    "text": "Issue 1217"
  }
]
```

The `system_name` is not one that is registered in RVISC.

6.3.21 Extension Tests

This subsubsection structures the informative tests for extensions. Each of the following tests SHOULD be treated as they where listed similar to the other tests.

An application MAY group these tests when providing the additional function to only run one or more selected tests.

6.3.21.1 Extension Category Critical

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that the extension `category` of the item is not `critical`.

It is sufficient to check whether the value of `category` is not `critical`.

The relevant paths for this test are:

```
/document/x_extensions[]/category
/product_tree/branches[](/branches[])*/product/x_extensions[]/category
/product_tree/full_product_names[]/x_extensions[]/category
/product_tree/product_paths[]/full_product_name/x_extensions[]/category
/vulnerabilities[]/metrics[]/content/x_extensions[]/category
/vulnerabilities[]/x_extensions[]/category
/x_extensions[]/category
```

Example 1 (which fails the test):

```
"x_extensions": [
  {
    // ...
    "category": "critical",
    // ...
  }
]
```

The extension category is `critical`. A reader that does not understand this extension might miss information that is critical to understand the content of the CSAF document.

6.3.21.2 Usage of Experimental Extension in Non TLP:CLEAR Document

For each item in an element of type `#/$defs/extensions_t` it MUST be tested that no experimental extension is used if the document is not labeled `TLP:CLEAR`.

The relevant paths for this test are:

```
/document/x_extensions[]
/product_tree/branches[](/branches[])*/product/x_extensions[]
/product_tree/full_product_names[]/x_extensions[]
/product_tree/product_paths[]/full_product_name/x_extensions[]
/vulnerabilities[]/metrics[]/content/x_extensions[]
/vulnerabilities[]/x_extensions[]
/x_extensions[]
```

Example 1 (which fails the test):

```
{
  // ...
  "document": {
    // ...
    "distribution": {
      "tlp": {
        "label": "GREEN"
      }
    },
    // ...
  }
  "x_extensions": [
    {
      "$schema": "https://raw.githubusercontent.com/oasis-tcs/csaf/refs/heads/master/
11/documentation-11-content_1.0.0.json",
      // ...
    }
  ]
}
```

The extension is an experimental CSAF Extension. Its properties and meaning might therefore not be known to the reader of the document.

6.3.21.3 Usage of Extension at Document Level

It MUST be tested that the element `/document/x_extensions` does not exist.

The relevant path for this test is:

```
/document/x_extensions
```

Example 1 (which fails the test):


```
{
  // ...
  "document": {
    // ...
    "x_extensions": [
      // ...
    ]
  }
}
```

The element `/document/x_extensions` exists.

6.3.21.4 Usage of Extension in Product Tree Branch Path

It MUST be tested that the element `x_extensions` does not exist in any path that starts with `/product_tree/branches`.

The relevant path for this test is:

```
/product_tree/branches[](/branches[])*/product/x_extensions
```

Example 1 (which fails the test):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              // ...
              "product": {
                // ...
                "x_extensions": [
                  // ...
                ]
              }
            }
          ],
          // ...
        }
      ],
      // ...
    }
  ]
}
```

The element `x_extensions` exists in a path that starts with `/product_tree/branches`.

6.3.21.5 Usage of Extension in Product Tree Full Product Names Path

It MUST be tested that the element `x_extensions` does not exist in any path that starts with `/product_tree/full_produc`

The relevant path for this test is:

```
/product_tree/full_product_names[]/x_extensions
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      // ...
      "x_extensions": [
        // ...
      ]
    }
  ]
}
```

The element `x_extensions` exists in a path that starts with `/product_tree/full_product_names`.

6.3.21.6 Usage of Extension in Product Tree Product Paths Path

It MUST be tested that the element `x_extensions` does not exist in any path that starts with `/product_tree/product_pat`

The relevant path for this test is:

```
/product_tree/product_paths[]/full_product_name/x_extensions
```

Example 1 (which fails the test):

```
"product_tree": {
  // ...
  "product_paths": [
    {
      // ...
      "full_product_name": {
        // ...
        "x_extensions": [
          // ...
        ]
      },
      // ...
    }
  ]
}
```

The element `x_extensions` exists in a path that starts with `/product_tree/product_paths`.

6.3.21.7 Usage of Extension in Vulnerabilities Metrics Path

It MUST be tested that the element `x_extensions` does not exist in any path that starts with `/vulnerabilities[]/metric`

The relevant path for this test is:

```
/vulnerabilities[]/metrics[]/content/x_extensions
```

Example 1 (which fails the test):

```
"vulnerabilities": [
  {
    "metrics": [
      {
        "content": {
          "x_extensions": [
            // ...
          ]
        },
        // ...
      }
    ]
  }
]
```

The element `x_extensions` exists in a path that starts with `/vulnerabilities[]/metrics`.

6.3.21.8 Usage of Extension at Vulnerabilities Level

For each item `/vulnerabilities` it MUST be tested that the element `x_extensions` does not exist.

The relevant path for this test is:

```
/vulnerabilities[]/x_extensions
```

Example 1 (which fails the test):

```
{
  // ...
  "vulnerabilities": [
    {
      // ...
      "x_extensions": [
        // ...
      ]
    }
  ]
}
```

The element `x_extensions` exists inside a `vulnerabilities` item.

6.3.21.9 Usage of Extension at Root Level

It MUST be tested that the element `/x_extensions` does not exist.

The relevant path for this test is:

```
/x_extensions
```

Example 1 (which fails the test):

```
{
  // ...
  "x_extensions": [
    // ...
  ]
}
```

The element `/x_extensions` exists.

6.3.22 Nested Product Path

For each item in `/product_tree/product_paths[]` it MUST be tested that all referenced Product IDs are not defined under a `full_product_name_t` element within items of `/product_tree/product_paths`.

A product defined by a product path that contains another product path is usually harder to match. Nevertheless, there are scenarios that require such a nesting to correctly describe the reality or provide additional matching information.

The relevant paths for this test are:

```
/product_tree/product_paths[]/beginning_product_reference
/product_tree/product_paths[]/subpaths[]/next_product_reference
```

Example 1 (which fails the test):

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-908070601"
    },
    {
      "name": "Product B",
      "product_id": "CSAFPID-908070602"
    },
    {
      "name": "Product C",
      "product_id": "CSAFPID-908070603"
    }
  ],
  "product_paths": [
    {
      "beginning_product_reference": "CSAFPID-908070601",
```



```
      "full_product_name": {
        "name": "Product A installed on Product B",
        "product_id": "CSAFPID-908070604"
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070602"
        }
      ]
    }
    {
      "beginning_product_reference": "CSAFPID-908070604",
      "full_product_name": {
        "name": "Product A installed on Product B installed on Product C",
        "product_id": "CSAFPID-908070605"
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070603"
        }
      ]
    }
  ]
}
```

The Product with the ID `CSAFPID-908070605` is defined through a product reference which belongs to a product formed by a product path.

6.4 Test Presets

A test preset is a predefined set of tests that was given a name. It MAY contain any number of tests. Two presets MAY overlap. The content of a preset MAY vary in different CSAF versions. A CSAF validator MUST support every official preset that solely include tests that are implemented by the CSAF validator. A CSAF validator MAY provide or support additional presets. A CSAF validator MUST implement all tests for any supported preset.

Names of presets not defined in this CSAF standard SHALL have the following prefix before their name:

- `x_`: for any CSAF validator specific preset.

  Multiple CSAF validators might use the same preset name for different sets of tests. Users are advised to carefully check the documentation of the tools to avoid incorrect assumptions.

- `org_` followed by an organization identifier and an underscore (_): for any preset specified by an organization as a part of a public definition that can be implemented by different CSAF validators. The organization identifier MUST only use the characters identified by the pattern `[0-9a-zA-Z-.]`. The organization identifier MUST be registered with the OASIS CSAF TC prior to the publication of the definition.
- `csaf_`: for any preset defined later on by the OASIS CSAF TC.

To avoid collisions especially in the `x_` namespace, the reverse DNS notation for a domain under the author's own control MAY be used.

Official presets are defined in different parts of the standard.

6.4.1 Presets Defined through Test Subsections

The following presets are defined through subsections of the test section:

- `mandatory`: all tests given in section 6.1
- `recommended`: all tests given in section 6.2
- `informative`: all tests given in section 6.3

6.4.2 Presets Defined through Conformance Targets

The following presets are defined through conformance targets:

- `schema`: Check against the JSON schema (see section 9.1.14)
- `basic`: `schema` + `mandatory` (see section 9.1.14)
- `extended`: `basic` + `recommended` (see section 9.1.15)
- `full`: `extended` + `informative` (see section 9.1.16)
- `additional`: all implemented tests that satisfy the CSAF Additional Test conformance profile (see section 9.1.27)

> Note: The last preset MAY vary between different implementations as it is implementation specific.

As presets are sets, the operator + MUST be interpreted as the union operation.

6.4.3 Additional Presets

Additional presets are defined as follows:

- `external-request-free`:
  - Description: Any test that can be executed without a request into the Internet or a different network.

    > This is intended to be used for browser-based tools as external requests may result in CORS issues. Request over network to a tool that is delivered with or an install requirement for a CSAF validator are not considered external.

  - Set: `full` excluding tests 6.3.6 and 6.3.7
- `consistent-revision-history`:
  - Description: Any test that is related to the revision history and ensures consistence within it.
  - Set:
    - 6.1.14
    - 6.1.18
    - 6.1.19
    - 6.1.21
    - 6.1.22
    - 6.1.37
    - 6.2.4
    - 6.2.5
    - 6.2.6
    - 6.2.21
    - 6.2.33
- `consistent-date-times`:
  - Description: Any test that is related to timestamps in CSAF and avoids invalid situations.

  - Set:
    - 6.1.37
    - 6.1.45
    - 6.1.49
    - 6.1.51
    - 6.1.52
    - 6.1.53

- `ssvc`:
  - Description: Any test that is related to SSVC in CSAF.
  - Set:
    - 6.1.46
    - 6.1.47
    - 6.1.48
    - 6.1.49
    - 6.2.3
    - 6.2.34
    - 6.2.35
    - 6.2.36
    - 6.2.37
    - 6.3.13
    - 6.3.14
    - 6.3.15

- `extensions`:
  - Description: Any test that is related to CSAF Extensions.
  - Set: -6.1.60.1 -6.1.60.2 -6.1.60.3 -6.2.39.4.1 -6.2.54.1 -6.2.54.2 -6.2.54.3 -6.2.54.4 -6.3.21.1 -6.3.21.2 -6.3.21.3 -6.3.21.4 -6.3.21.5 -6.3.21.6 -6.3.21.7 -6.3.21.8 -6.3.21.9

- `extensions-exist`:
  - Description: Detects CSAF Extensions in all places.
  - Set: -6.3.21.3 -6.3.21.4 -6.3.21.5 -6.3.21.6 -6.3.21.7 -6.3.21.8 -6.3.21.9

7 Distributing CSAF Documents

This section lists requirements and roles defined for distributing CSAF documents. The first subsection provides all requirements - the second one the roles. It is mandatory to fulfill the basic role “CSAF Publisher”. The last section provides specific rules for the process of retrieving CSAF documents.

7.1 Requirements

The requirements in this subsection are consecutively numbered to be able to refer to them directly. The order does not give any hint about the importance. Not all requirements have to be fulfilled to conform to this specification - the sets of requirements per conformance clause are defined in section 7.2.

7.1.1 Requirement 1: Valid CSAF Document

The document is a valid CSAF document (cf. Conformance clause 1).

7.1.2 Requirement 2: Filename

The CSAF document has a filename according to the rules in section 5.1.

7.1.3 Requirement 3: TLS

The CSAF document is per default retrievable from a website which uses TLS for encryption and server authenticity. The CSAF document MUST NOT be downloadable from a location which does not encrypt the transport when crossing organizational boundaries to maintain the chain of custody.

7.1.4 Requirement 4: TLP:CLEAR

If the CSAF document is labeled TLP:CLEAR, it MUST be freely accessible.

This does not exclude that such a document is also available in an access protected customer portal. However, there MUST be one copy of the document available for people without access to the portal.

> Reasoning: If an advisory is already in the media, an end user should not be forced to collect the pieces of information from a press release but be able to retrieve the CSAF document.

7.1.5 Requirement 5: TLP:AMBER, TLP:AMBER+STRICT and TLP:RED

CSAF documents labeled TLP:AMBER, TLP:AMBER+STRICT or TLP:RED MUST be access protected. If they are provided via a web server this SHALL be done under a different path than for TLP:CLEAR, TLP:GREEN and unlabeled CSAF documents. TLS client authentication, access tokens or any other automatable authentication method SHALL be used.

An issuing party MAY agree with the recipients to use any kind of secured drop at the recipients' side to avoid putting them on their own website. However, it MUST be ensured that the documents are still access protected.

7.1.6 Requirement 6: No Redirects

Redirects SHOULD NOT be used. If they are inevitable only HTTP Header redirects are allowed.

> Reasoning: Clients should not parse the payload for navigation and some, as e.g. `curl`, do not follow any other kind of redirects.

If any redirects are used, there SHOULD NOT be more than `10`, and MUST NOT be more than `20` consecutive redirects.

> This aligns with section 4.4 of the [FETCH] specification.

7.1.7 Requirement 7: provider-metadata.json

The party MUST provide a valid `provider-metadata.json` according to the schema CSAF provider metadata for its own metadata. The `publisher` object SHOULD match the one used in the CSAF documents of the issuing party but can be set to whatever value a CSAF aggregator SHOULD display over any individual `publisher` values in the CSAF documents themselves.

> This information is used to collect the data for CSAF aggregators, listers and end users. The CSAF provider metadata schema ensures the consistency of the metadata for a CSAF provider across the ecosystem. Other approaches, like extracting the `publisher` object from CSAF documents, are likely to fail if the object differs between CSAF documents.
> It is suggested to put the file `provider-metadata.json` adjacent to the ROLIE feed documents (requirement 15) or in the main directory adjacent to the year folders (requirement 14), `changes.csv` (requirement 13) and the `index.txt` (requirement 12). Suggested locations to store the `provider-metadata.json` are:
>
> - https://www.example.com/.well-known/csaf/provider-metadata.json
> - https://domain.tld/security/data/csaf/provider-metadata.json
> - https://psirt.domain.tld/advisories/csaf/provider-metadata.json
> - https://domain.tld/security/csaf/provider-metadata.json

Example 1 (minimal with ROLIE document):

```
{
  "$schema": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/provider.json",
  "canonical_url": "https://www.example.com/.well-known/csaf/provider-metadata.json",
  "distributions": [
    {
      "rolie": {
        "feeds": [
          {
            "last_updated": "2024-01-24T20:20:56.169Z",
            "summary": "All TLP:CLEAR advisories of Example Company.",
            "tlp_label": "CLEAR",
            "url": "https://www.example.com/.well-known/csaf/feed-tlp-clear.json"
          }
        ]
      }
    }
  ],
  "last_updated": "2024-01-24T20:20:56.169Z",
  "list_on_CSAF_aggregators": true,
  "metadata_version": "2.1",
  "mirror_on_CSAF_aggregators": true,
  "public_openpgp_keys": [
    {
      "fingerprint": "8F5F267907B2C4559DB360DB2294BA7D2B2298B1",
      "url": "https://keys.example.net/vks/v1/by-fingerprint/8F5F267907B2C4559DB360DB
    }
  ],
  "publisher": {
    "category": "vendor",
    "name": "Example Company ProductCERT",
    "namespace": "https://psirt.example.com"
  },
  "role": "csaf_trusted_provider"
}
```

The `maintained_until` and `maintained_from` properties can be used to indicate that the distributions contained in `provider-metadata.json` at the given `canonical_url` are only guaranteed to be maintained until or after the specified date and time. This SHOULD be used to support a transition period between CSAF 2.0 and CSAF 2.1 (cf. section 7.4). CSAF downloaders (cf. section 9.1.23) and programs retrieving or providing a `provider-metadata.json` SHOULD evaluate this property and emit a warning if the current date is less than 90 days away from `maintained_until` and an error if the current date exceeds `maintained_until`. The programs MAY provide a non-default option to use the `provider-metadata.json` anyway. Furthermore, such programs SHOULD evaluate the `maintained_from` property and output a warning if the current date is still before the `maintained_from` timestamp.

If a CSAF publisher (cf. section 7.2.1) does not provide the `provider-metadata.json`, an aggregator SHOULD contact the CSAF publisher in question to determine the values for `list_on_CSAF_aggregators` and `mirror_on_CSAF_aggregators`. If that is impossible or if the CSAF publisher is unresponsive the following values MUST be used:

```
  "list_on_CSAF_aggregators": true,
  "mirror_on_CSAF_aggregators": false
```


> This prevents that CSAF documents of a CSAF publisher which have been collected by one CSAF aggregator A are mirrored again on a second CSAF aggregator B. Such cascades are prone to outdated information. If the first aggregator A collects the CSAF documents on best effort and B copies the files from A and announces that this is done weekly, one might assume that B's CSAF documents are more recent. However, that is not the case as B's information depends on A.

7.1.8 Requirement 8: security.txt

In the security.txt there MUST be at least one field `CSAF` which points to the `provider-metadata.json` (requirement 7). If this field indicates a web URI, then it MUST begin with "https://" (as per section 2.7.2 of [RFC7230]). See [SECURITY-TXT] for more details.

> The security.txt was published as [RFC9116] in April 2022. The `CSAF` field was officially added through the IANA registry.

Examples 1:

```
CSAF: https://www.example.com/.well-known/csaf/provider-metadata.json
CSAF: https://domain.tld/security/data/csaf/provider-metadata.json
CSAF: https://psirt.domain.tld/advisories/csaf/provider-metadata.json
CSAF: https://domain.tld/security/csaf/provider-metadata.json
```

It is possible to advertise more than one `provider-metadata.json` by adding multiple `CSAF` fields, e.g. in case of changes to the organizational structure through mergers and acquisitions. However, this SHOULD NOT be done and removed as soon as possible. A valid use case for temporarily including multiple entries would be a transition phase between different CSAF versions, in which documents and provider metadata of both versions are served simultaneously (cf. section 7.4). If one of the URLs fulfills requirement 9, it MUST be set as the first CSAF entry in the security.txt.

7.1.9 Requirement 9: Well-Known URL for provider-metadata.json

The URL path `/.well-known/csaf/provider-metadata.json` under the main domain of the issuing authority serves directly the `provider-metadata.json` according to requirement 7. That implies that redirects SHALL NOT be used. The use of the scheme "HTTPS" is required. See [RFC8615] for more details.

Example 1:

```
https://www.example.com/.well-known/csaf/provider-metadata.json
```

As specified in 7.4, the value of `canonical_url` MAY differ from the URL that was requested as a part of this requirement. Such state is intended and MUST NOT be reported as error.

7.1.10 Requirement 10: DNS Path

Assuming that the organization's main domain is `domain.tld`, the DNS record `csaf.data.security.domain.tld` SHALL resolve to the IP address of a web server which serves directly the `provider-metadata.json` according to requirement 7.

> The `domain.tld` is just a placeholder for the organization's main domain. For the organization with the main domain being `example.com`, the necessary DNS record is `csaf.data.security.example.com`.

That implies that redirects SHALL NOT be used. The use of the scheme "HTTPS" is required.

7.1.11 Requirement 11: One Folder per Year

The CSAF documents MUST be located within folders named `<YYYY>` where `<YYYY>` is the year given in the value of `/document/tracking/initial_release_date`.

Examples 1:

```
2024
2023
```

7.1.12 Requirement 12: index.txt

The file `index.txt` MUST contain the list of all filenames of CSAF documents which are located in the sub-directories with their filenames. Each entry SHALL be terminated by a newline sequence. The last entry MAY skip the newline sequence.

Example 1:

```
2025/esa-2025-421324.json
2024/esa-2024-620109.json
2024/esa-2024-470520.json
2024/esa-2024-430524.json
2023/esa-2023-450116.json
```

> This can be used to download all CSAF documents.

The file `index.txt` SHALL be located in the folder given as directory URL under `/distributions[]/directory/url` in the `provider-metadata.json`.

> If different TLP labels are used, multiple `index.txt` exist. However, they are located in the corresponding folders and contain only the filenames of files for that TLP label.

Example 2:

```
.well-known/
├─ csaf/
│   ├─ amber/
│   │   ├─ 2023/
│   │   │   ├─ esa-2023-540205.json
│   │   ├─ 2025/
│   │   │   ├─ esa-2025-440412.json
│   │   ├─ changes.csv
│   │   ├─ index.txt
│   ├─ amber+strict/
│   │   ├─ 2024/
│   │   │   ├─ esa-2024-451013.json
│   │   ├─ 2025/
│   │   │   ├─ esa-2025-540204.json
│   │   ├─ changes.csv
│   │   ├─ index.txt
│   ├─ clear/
│   │   ├─ 2023/
```



```
│   │   │   ├─ esa-2023-450116.json
│   │   ├─ 2024/
│   │   │   ├─ esa-2024-430524.json
│   │   │   ├─ esa-2024-470520.json
│   │   │   ├─ esa-2024-620109.json
│   │   ├─ 2025/
│   │   │   ├─ esa-2025-421324.json
│   │   ├─ changes.csv
│   │   ├─ index.txt
│   ├─ green/
│   │   ├─ 2024/
│   │   │   ├─ esa-2024-430316.json
│   │   ├─ 2025/
│   │   │   ├─ esa-2025-431009.json
│   │   ├─ changes.csv
│   │   ├─ index.txt
│   ├─ red/
│   │   ├─ 2022/
│   │   │   ├─ esa-2022-431406.json
│   │   │   ├─ esa-2022-431407.json
│   │   ├─ changes.csv
│   │   ├─ index.txt
```

The example [[2]] uses five `index.txt` files - one for each TLP label. The corresponding `provider-metadata.json` excerpt is given in example [[3]]. Example [[1]] depicts the content of the `index.txt` within the folder `clear` located at `https://www.example.com/.well-known/csaf/clear/index.txt`.

Example 3:

```
  "distributions": [
    {
      "directory": {
        "tlp_label": "CLEAR",
        "url": "https://www.example.com/.well-known/csaf/clear/",
      }
    },
    {
      "directory": {
        "tlp_label": "GREEN",
        "url": "https://www.example.com/.well-known/csaf/green/",
      }
    },
    {
      "directory": {
        "tlp_label": "AMBER",
        "url": "https://www.example.com/.well-known/csaf/amber/",
      }
    },
    {
```



```
      "directory": {
        "tlp_label": "AMBER+STRICT",
        "url": "https://www.example.com/.well-known/csaf/amber+strict/",
      }
    },
    {
      "directory": {
        "tlp_label": "RED",
        "url": "https://www.example.com/.well-known/csaf/red/",
      }
    }
  ],
```

7.1.13 Requirement 13: changes.csv

The file `changes.csv` contains a list of CSAF documents in the current TLP level that were changed recently. Therefore, it MUST contain the filename as well as the value of `/document/tracking/current_release_date` for each CSAF document in the sub-directories without a heading; lines MUST be sorted by the `current_release_date` timestamp with the latest one first. The `changes.csv` SHALL be a valid comma separated values format as defined by [RFC4180] without double quotes.

> Note: As a consequence of section 7.1.2 Requirement 2 for filenames and section 7.1.11 Requirement for directory names, there must not be any characters within the `changes.csv` that would require quoting.

Example 1:

```
2024/esa-2024-430524.json,2025-07-21T11:14:37Z
2025/esa-2025-421324.json,2025-01-10T00:51:10Z
2024/esa-2024-470520.json,2025-01-01T15:34:54Z
2023/esa-2023-450116.json,2024-12-01T17:09:02Z
2024/esa-2024-620109.json,2024-07-30T23:59:29Z
```

> Note: As CSAF 2.0 requires quotes, an [RFC4180] parser can read both format revisions.

The file `changes.csv` SHALL be located in the folder given as directory URL under `/distributions[]/directory/url` in the `provider-metadata.json`.

> The example [[2 (of section 7.1.12)]] uses five `changes.csv` files - one for each TLP label. The corresponding `provider-metadata.json` excerpt is given in example [[3 (of section 7.1.12)]]. Example [[1]] depicts the content of the `changes.csv` within the folder `clear` located at `https://www.example.com/.well-known/csaf/clear/changes.csv`.

7.1.14 Requirement 14: Directory Listings

Server-side generated directory listing SHALL be enabled to support manual navigation.

> As the content of the directory listing is more or less static, there is little to no benefit in using of client-side scripts. Moreover, client-side scripts, like JavaScript, are usually not evaluated in text-based browsers and are also hard to check programmatically.

7.1.15 Requirement 15: ROLIE Feed

Resource-Oriented Lightweight Information Exchange (ROLIE) is a standard to ease discovery of security content. ROLIE is built on top of the Atom Publishing Format and Protocol, with specific requirements that support publishing security content. All CSAF documents with the same TLP level MUST be listed in a single ROLIE feed (summary feed). Additional ROLIE feeds might exist that contain only a subset of the CSAF documents. The selection criteria SHOULD be described through the summary. At least one of the feeds

- TLP:CLEAR
- TLP:GREEN

MUST exist. Each ROLIE feed document MUST be a JSON file that conforms with [RFC8322].

The ROLIE feed document MUST contain a feed category with the registered ROLIE information type `csaf`. The `scheme` for this category MUST be `urn:ietf:params:rolie:category:information-type`.

Example 1:

```
{
  "feed": {
    "id": "example-csaf-feed-tlp-clear",
    "title": "Example CSAF feed (TLP:CLEAR)",
    "link": [
      {
        "rel": "self",
        "href": "https://psirt.domain.tld/advisories/csaf/feed-tlp-clear.json"
      }
    ],
    "category": [
      {
        "scheme": "urn:ietf:params:rolie:category:information-type",
        "term": "csaf"
      }
    ],
    "updated": "2024-01-01T12:00:00.000Z",
    "entry": [
      {
        "id": "ESA-2024-001",
        "title": "Multiple vulnerabilities in ABC 0.0.2",
        "link": [
          {
            "rel": "self",
            "href": "https://psirt.domain.tld/advisories/csaf/2024/esa-2024-001.json"
          },
          {
            "rel": "hash",
            "href": "https://psirt.domain.tld/advisories/csaf/2024/esa-2024-001.json.
          },
          {
            "rel": "signature",
            "href": "https://psirt.domain.tld/advisories/csaf/2024/esa-2024-001.json.
          }
        ],
        "published": "2024-01-01T11:00:00.000Z",
```



```
          "updated": "2024-01-01T12:00:00.000Z",
          "summary": {
            "content": "Multiple vulnerabilities were fixed in ABC 0.0.3"
          },
          "content": {
            "type": "application/json",
            "src": "https://psirt.domain.tld/advisories/csaf/2024/esa-2024-001.json"
          },
          "format": {
            "schema": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/csaf.json",
            "version": "2.1"
          }
        }
      ]
    }
  }
```

Any existing hash file (requirement 18) MUST be listed in the corresponding entry of the ROLIE feed as an item of the array `link` having the `rel` value of `hash`. Any existing signature file (requirement 19) MUST be listed in the corresponding entry of the ROLIE feed as an item of the array `link` having the `rel` value of `signature`.

7.1.16 Requirement 16: ROLIE Service Document

The use and therefore the existence of ROLIE service document is optional. If it is used, each ROLIE service document MUST be a JSON file that conforms with [RFC8322] and lists the ROLIE feed documents. Additionally, it can also list the corresponding ROLIE category documents. The ROLIE service document SHOULD use the filename `service.json` and reside next to the `provider-metadata.json`.

Example 1:

```
{
  "service": {
    "workspace": [
      {
        "title": "Public CSAF feed",
        "collection": [
          {
            "title": "Example CSAF feed (TLP:CLEAR)",
            "href": "https://psirt.domain.tld/advisories/csaf/feed-tlp-clear.json",
            "categories": {
              "category": [
                {
                  "scheme": "urn:ietf:params:rolie:category:information-type",
                  "term": "csaf"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```


7.1.17 Requirement 17: ROLIE Category Document

The use and therefore the existence of ROLIE category document is optional. If it is used, each ROLIE category document MUST be a JSON file that conforms with [RFC8322]. A ROLIE category document SHOULD reside next to the corresponding ROLIE feed. ROLIE categories SHOULD be used for to further dissect CSAF documents by one or more of the following criteria:

- document category
- document language
- values of the branch category within the Product Tree including but not limited to

  ```
  - vendor
  - product_family
  - product_name
  - product_version
  ```

- type of product

  Examples 1:

  ```
  CPU
  Firewall
  Monitor
  PLC
  Printer
  Router
  Sensor
  Server
  ```

- areas or sectors, the products are used in

  Examples 2:

  ```
  Chemical
  Commercial
  Communication
  Critical Manufacturing
  Dams
  Energy
  Healthcare
  Water
  ```

- any other categorization useful to the consumers

Example 3:

```
{
  "categories": {
    "category": [
      {
          "term": "Example Company Product A"
      },
      {
          "term": "Example Company Product B"
      }
```



```
      ]
    }
  }
```

7.1.18 Requirement 18: Integrity

All CSAF documents SHALL have at least one hash file computed with a secure cryptographic hash algorithm (e.g. SHA-512 or SHA-3) to ensure their integrity. The filename is constructed by appending the file extension which is given by the algorithm.

MD5 and SHA1 SHALL NOT be used.

Example 1:

```
File name of CSAF document: esa-2022-02723.json
File name of SHA-256 hash file: esa-2022-02723.json.sha256
File name of SHA-512 hash file: esa-2022-02723.json.sha512
```

The file content SHALL start with the first byte of the hexadecimal hash value. The hash value SHALL be represented in lowercase. Any subsequent data (like a filename) which is optional SHALL be separated by at least one space.

Example 2:

```
ea6a209dba30a958a78d82309d6cdcc6929fcb81673b3dc4d6b16fac18b6ff38  esa-2022-02723.json
```

If a ROLIE feed exists, each hash file MUST be listed in it as described in requirement 15.

7.1.19 Requirement 19: Signatures

All CSAF documents SHALL have at least one OpenPGP signature file which is provided under the same filename which is extended by the appropriate extension. This signature SHALL be presented as an ASCII armored file. See [RFC4880] for more details.

Example 1:

```
File name of CSAF document: esa-2022-02723.json
File name of signature file: esa-2022-02723.json.asc
```

If a ROLIE feed exists, each signature file MUST be listed in it as described in requirement 15.

At all times, signatures MUST remain valid for a minimum of 30 days and ideally for at least 90 days. When executing CSAF document signatures, the signing party SHOULD adhere to or surpass the prevailing best practices and recommendations regarding key length. Tools SHOULD treat the violation of the rules given in the first sentence as:

- warning if the signature is only valid for 90 days or less at the time of the verification,
- error, which MAY be ignored by the user per option, if the signature is only valid for 30 days or less at the time of the verification and
- error if the signature is expired at the time of the verification.

7.1.20 Requirement 20: Public OpenPGP Key

The public part of the OpenPGP key used to sign the CSAF documents MUST be available. This key file SHALL be presented as an ASCII armored file. It SHOULD also be available at a public key server.

> For example, the public part of the OpenPGP key could be placed in a directory `openpgp` adjacent to the `provider-metadata.json`.

The OpenPGP key SHOULD have a strength that is considered secure.

> Guidance on OpenPGP key strength can be retrieved from technical guidelines of competent authorities.

7.1.21 Requirement 21: List of CSAF Providers

The file `aggregator.json` MUST be present and valid according to the JSON schema CSAF aggregator. It MUST NOT be stored adjacent to a `provider-metadata.json`.

The file `aggregator.json` SHOULD be accessible at the registered path in the `.well-known` directory: `/.well-known/csaf-aggregator/aggregator.json`.

Examples 1:

```
https://aggregator.example/.well-known/csaf-aggregator/aggregator.json
https://aggregator.example/.well-known/csaf-aggregator/v2.1/aggregator.json
```

The file `aggregator.json` SHOULD only list the latest version of the metadata of a CSAF provider.

Example 2:

```
{
  "$schema": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/aggregator.json",
  "aggregator": {
    "category": "lister",
    "contact_details": "Example CSAF Lister can be reached at contact_us@lister.examp
    "issuing_authority": "This service is provided as it is. It is free for everybody
    "name": "Example CSAF Lister",
    "namespace": "https://lister.example"
  },
  "aggregator_version": "2.1",
  "canonical_url": "https://aggregator.example/.well-known/csaf-aggregator/aggregator
  "csaf_providers": [
    {
      "metadata": {
        "last_updated": "2024-01-12T20:20:56.169Z",
        "publisher": {
          "category": "vendor",
          "name": "Example Company ProductCERT",
          "namespace": "https://psirt.example.com"
        },
        "role": "csaf_provider",
        "url": "https://www.example.com/.well-known/csaf/provider-metadata.json"
      }
    },
    {
      "metadata": {
        "last_updated": "2024-01-12T21:35:38.000Z",
        "publisher": {
          "category": "coordinator",
```



```
            "name": "Example Coordinator CERT",
            "namespace": "https://cert.example"
          },
          "role": "csaf_trusted_provider",
          "url": "https://cert.example/advisories/csaf/provider-metadata.json"
        }
      }
    ],
    "last_updated": "2024-01-24T22:35:38.978Z"
  }
```

7.1.22 Requirement 22: Two Disjoint Issuing Parties

The file `aggregator.json` (requirement 21) lists at least two disjoint CSAF providers (including CSAF trusted providers) or one CSAF publisher and one CSAF provider (including CSAF trusted provider).

7.1.23 Requirement 23: Mirror

The CSAF documents for each issuing party that is mirrored MUST be in a different folder. The folder name SHOULD be retrieved from the name of the issuing authority. This folders MUST be adjacent to the `aggregator.json` (requirement 21). Each such folder MUST at least:

- provide a `provider-metadata.json` for the current issuing party.
- provide the ROLIE feed document according to requirement 15 which links to the local copy of the CSAF document.

Example 1:

```
  {
    "$schema": "https://docs.oasis-open.org/csaf/csaf/v2.1/schema/aggregator.json",
    "aggregator": {
      "category": "aggregator",
      "contact_details": "Example Aggregator can be reached at contact_us@aggregator.ex
      "issuing_authority": "This service is provided as it is. It is free for everybody
      "name": "Example Aggregator",
      "namespace": "https://aggregator.example"
    },
    "aggregator_version": "2.1",
    "canonical_url": "https://aggregator.example/.well-known/csaf-aggregator/aggregator
    "csaf_providers": [
      {
        "metadata": {
          "last_updated": "2024-01-12T20:20:56.169Z",
          "publisher": {
            "category": "vendor",
            "name": "Example Company ProductCERT",
            "namespace": "https://psirt.example.com"
          },
          "role": "csaf_provider",
          "url": "https://www.example.com/.well-known/csaf/provider-metadata.json"
        },
        "mirrors": [
          "https://aggregator.example/.well-known/csaf-aggregator/Example_Company_Produ
metadata.json"
```



```
        ]
      },
      {
        "metadata": {
          "last_updated": "2024-01-12T21:35:38.000Z",
          "publisher": {
            "category": "coordinator",
            "name": "Example Coordinator CERT",
            "namespace": "https://cert.example"
          },
          "role": "csaf_trusted_provider",
          "url": "https://cert.example/advisories/csaf/provider-metadata.json"
        },
        "mirrors": [
          "https://aggregator.example/.well-known/csaf-aggregator/Example_Coordinator_C
metadata.json"
        ]
      }
    ],
    "last_updated": "2024-01-24T22:35:38.978Z"
  }
```

7.1.24 Requirement 24: HTTP User-Agent

Access to the CSAF related files and directories provided, for both metadata and documents, MUST be allowed independent of the value of HTTP User-Agent.

> Limit the value of HTTP User-Agents to a certain set would hinder adoption of tools retrieving the files.

The only exception is that the temporary blocking of certain HTTP User-Agents is allowed to mitigate an ongoing security incident (e.g. a DoS attack on the web server serving the CSAF files). However, a less severe measure with a similar effect SHOULD be used. CSAF related files and directories SHOULD be exempted from temporary blocking. The temporary blocking SHOULD be removed as soon as possible, at latest two weeks after the security incident process was completed.

> Also confer to the TC's guidance on content delivery networks and caching.

7.1.25 Requirement 25: Access-Control-Allow-Origin

For any CSAF documents and related metadata, the web server SHOULD set the HTTP header `Access-Control-Allow-Origin: *`.

> The HTTP header enables users to access the CSAF data with web browser based clients.

The value of the HTTP header MAY be altered to allow just specified domains. In such case, the response SHOULD follow the recommendations of [FETCH] including but not limited to those about the `Vary` header.

> Such restriction may allow the allow-listed domains to send credentials.

7.2 Roles

This subsection groups the requirements from the previous subsection into named sets which target the roles with the same name. This allows end users to request their suppliers to fulfill a certain set of requirements. A supplier can use roles for advertising and marketing.

The roles “CSAF Publisher”, “CSAF Provider”, and “CSAF Trusted Provider” are intended directly for issuing parties and form the first group. The second group consists of the roles “CSAF Lister” and “CSAF Aggregator”. They collect data from the aforementioned issuing parties of the first group and provide them in a single place to aid in automation. Parties of the second group can also issue their own advisories. However, they MUST follow the rules for the first group for that.

Both, a CSAF lister and a CSAF aggregator, decide based on their own rules which issuing parties to list respectively to mirror. However, an issuing party MAY apply to be listed or mirrored.

Issuing parties MUST indicate through the value `false` in `list_on_CSAF_aggregators` if they do not want to be listed. Issuing parties MUST indicate through the value `false` in `mirror_on_CSAF_aggregators` if they do not want to be mirrored.

The values are independent. The combination of the value `false` in `list_on_CSAF_aggregators` and `true` in `mirror_on_CSAF_aggregators` implies that the issuing party does not want to be listed without having the CSAF documents mirrored. Therefore, a CSAF aggregator can list that issuing party if it mirrors the files.

7.2.1 Role: CSAF Publisher

A distributing party satisfies the “CSAF Publisher” role if the party:

- satisfies the requirements 1 to 4 in section 7.1.
- distributes only CSAF documents on behalf of its own.

7.2.2 Role: CSAF Provider

A CSAF publisher satisfies the “CSAF Provider” role if the party fulfills the following three groups of requirements:

Firstly, the party:

- satisfies the “CSAF Publisher” role profile.
- additionally satisfies the requirements 5 to 7, 24 and 25 in section 7.1.

Secondly, the party:

- satisfies at least one of the requirements 8 to 10 in section 7.1.

Thirdly, the party:

- satisfies the requirements 11 to 14 in section 7.1 or requirements 15 to 17 in section 7.1.

> If the party uses the ROLIE-based distribution, it MUST also satisfy requirements 15 to 17. If it uses the directory-based distribution, it MUST also satisfy requirements 11 to 14.

7.2.3 Role: CSAF Trusted Provider

A CSAF provider satisfies the “CSAF Trusted Provider” role if the party:

- satisfies the “CSAF Provider” role profile.
- additionally satisfies the requirements 18 to 20 in section 7.1.

7.2.4 Role: CSAF Lister

A distributing party satisfies the "CSAF Lister" role if the party:

- satisfies the requirements 6, 21, 22, 24 and 25 in section 7.1.
- uses the value `lister` for `/aggregator/category`.
- does not list any mirror pointing to a domain under its own control.

> The purpose of this role is to provide a list of URLs where to find CSAF documents. It is not assumed that the list will be complete.

7.2.5 Role: CSAF Aggregator

A distributing party satisfies the "CSAF Aggregator" role if the party:

- satisfies the requirements 1 to 6 and 21 to 25 in section 7.1.
- uses the value `aggregator` for `/aggregator/category`.
- lists a mirror for at least two disjoint issuing parties pointing to a domain under its own control.
- links the public part of the OpenPGP key used to sign CSAF documents for each mirrored issuing party in the corresponding `provider-metadata.json`.
- provides for each CSAF document that is mirrored a signature (requirement 19) and a hash (requirement 18). Both SHALL be listed in the ROLIE feed. If the issuing party provides those files for a CSAF document, they SHOULD be copied as well. If the issuing party does not provide those files, they SHALL be created by the CSAF aggregator. Such a signature does not imply any liability of CSAF aggregator for the content of the corresponding CSAF document. It just confirms that the CSAF document provided has not been modified after being downloaded from the issuing party. A CSAF aggregator MAY add additional signatures and hashes for a CSAF document.

Additionally, a CSAF aggregator MAY list one or more issuing parties that it does not mirror.

> The purpose of this role is to provide a single point where CSAF documents can be retrieved. Multiple CSAF aggregators are expected to exist around the world. None of them is required to mirror all CSAF documents of all issuing parties. CSAF aggregators can be provided for free or as a paid service.
> To aid in automation, CSAF aggregators MAY mirror CSAF documents from CSAF publishers. Regarding the terms of use they SHOULD consult with the issuing party. The purpose of this option is that a consumer can retrieve CSAF documents from a CSAF publisher as if this issuing party would be a CSAF trusted provider. To reach that goal, a CSAF aggregator collects the CSAF documents from the CSAF publisher and mirrors it. The collection process MAY be automated or manual. CSAF aggregators announce the collection interval through the field `update_interval` in the corresponding item of the CSAF publishers list (`csaf_publishers`) in their `aggregator.json`. To minimize the implementation efforts and process overhead, a CSAF aggregator MAY upload the CSAF documents of a CSAF publisher into an internal instance of a CSAF provider software. Such construct is called "CSAF Proxy Provider" as it can be mirrored by the CSAF aggregator software. However, such a CSAF proxy provider MUST NOT be accessible from anyone else than the CSAF aggregator itself. Otherwise, that would violate the second rule of section 7.2.1. Therefore, it is recommended to expose the CSAF proxy provider only on localhost and allow the access only from the CSAF aggregator software.

7.3 Retrieving Rules

The retrieving process executes in two phases: Finding the `provider-metadata.json` (requirement 7 in section 7.1) and retrieving CSAF documents.

> A retrieving party SHOULD do the first phase every time. Based on the setup and use case of the retrieving party it MAY choose to do it less often, e.g. only when adding new or updating distributing parties. In that case, it SHOULD to check regularly whether new information is available.

7.3.1 Finding provider-metadata.json

Direct locating: The following process SHOULD be used to determine the location of a `provider-metadata.json` (requirement 7 in section 7.1) based on the main domain of the issuing party.

First, an ordered list of possible `provider-metadata.json` candidates SHOULD be generated in the following way:

1. Checking the Well-known URL (requirement 9 in section 7.1)
2. Checking the security.txt (requirement 8 in section 7.1)
3. If the above steps fail to produce any candidates: Checking the DNS path (requirement 10 in section 7.1)

Second, select one or more `provider-metadata.json` to use from the list of valid candidates. If the retrieving party is only able to process one `provider-metadata.json`, the first one in the list SHOULD be chosen.

> The term "checking" used in the listing above SHOULD be understood as follows: Try to access the resource and test whether the response provides an expected result as defined in the respective requirement of that step as defined in section 7.1. If that is the case, the response is added to the list of candidates - otherwise not. If the resource yields more than one response, the responses are added to the list in the order they are returned from the resource.

Indirect locating: A retrieving party MAY choose to determine the location of a `provider-metadata.json` by retrieving its location from an `aggregator.json` (requirement 21 in section 7.1) of a CSAF lister or CSAF aggregator.

7.3.2 Retrieving CSAF Documents

Given a `provider-metadata.json`, the following process SHOULD be used to retrieve CSAF documents:

1. Parse the `provider-metadata.json` to determine whether the directory-based (requirements 11 to 14 in section 7.1) or ROLIE-based distribution (requirements 15 to 17 in section 7.1) is used. If both are present, the ROLIE information SHOULD be preferred.
2. For any CSAF trusted provider, the hash and signature files (requirements 18 to 19 in section 7.1) SHOULD be retrieved together with the CSAF document. They MUST be checked before further processing the CSAF document.
3. Test the CSAF document against the schema.
4. Execute mandatory tests on the CSAF document.

7.3.3 Finding aggregator.json

Direct locating: The file `aggregator.json` SHOULD be located at the registered `.well-known` path based on the main domain of the aggregator as specified in requirement 21 in section 7.1.

Indirect locating: A retrieving party MAY choose to determine the location of an `aggregator.json` through an out-of-band channel, e.g. an email list, news articles or a public website.

7.4 Transition between CSAF 2.0 and CSAF 2.1

This subsection details the process that SHOULD be followed when transitioning the distribution of documents from CSAF 2.0 to CSAF 2.1. Different scenarios can be encountered:

- Providers will continue to use CSAF 2.0. This SHOULD be avoided, as the CSAF ecosystem greatly profits from adoption of the new standard version.
- Providers will immediately upgrade all documents as well as the `provider-metadata.json` to CSAF 2.1. While this benefits the adoption of CSAF 2.1, consumers that still rely on CSAF 2.0 are cut off.
- Providers will begin a transition period, in which they continue to serve existing documents in CSAF 2.0, gradually updating the existing document base (e.g. by using a CSAF 2.0 to CSAF 2.1 converter as described in 9.1.18) and publishing new documents using CSAF 2.1.

7.4.1 Announcing the Transition

In the last scenario, a temporary parallel distribution of CSAF 2.0 and CSAF 2.1 documents and provider metadata is RECOMMENDED. The provider SHOULD announce a transition period containing three points in time:

- The beginning of the transition period, where the provider is starting to serve CSAF 2.1 documents, while CSAF 2.0 being authoritative.

  > It is expected that the CSAF 2.1 files can be used in production from this point onward.

- The roll-over-date at which CSAF 2.1 becomes authoritative but CSAF 2.0 is still supported.
- The end of the transition period, after which CSAF 2.0 is not supported, i.e. maintained, any more.

The announcement MAY contain also the following information:

- The first date, when CSAF 2.1 documents are available and automatically retrievable from the server through a CSAF 2.1 `provider-metadata.json`.

  > This might be a public beta prior to the begin of the transition period. However, those documents SHOULD NOT be used in production as they may be unstable due to their beta status.

- The last date, when CSAF 2.0 documents are available and automatically retrievable from the server through a CSAF 2.0 `provider-metadata.json`.

  > This date is usually at the end of the transition period.

7.4.2 Transition Process for a CSAF Provider

The following process SHOULD be followed:

- A `provider-metadata.json` in conformance to CSAF 2.0 SHOULD be placed at `/.well-known/csaf/v2.0/provider-metadata.json`.
  - Its `canonical_url` MUST be set to an URL corresponding to the `/.well-known/csaf/v2.0/provider-metadata.json` path.

    > The property `maintained_until` was not defined in CSAF 2.0 and therefore cannot be used - otherwise it would be set to the end of the transition period.

  - The URL SHALL also be added as an entry in the security.txt, if used.
- A `provider-metadata.json` in conformance to CSAF 2.1 SHOULD be placed at `/.well-known/csaf/v2.1/provider-metadata.json`.
  - Its `canonical_url` MUST be set to an URL corresponding to the `/.well-known/csaf/v2.1/provider-metadata.json` path.
  - Optionally, the property `maintained_from` can be set to an appropriate date in the future, if the service is not considered stable yet.

    > The transition officially starts at the date and time noted in the property `maintained_from` as this is the point in time when both CSAF 2.0 and CSAF 2.1 documents are retrievable in production-grade quality.

  - The URL SHALL also be added to the security.txt, if used.
- At the begin of the transition period, a `provider-metadata.json` in conformance to CSAF 2.0 SHOULD be placed at `/.well-known/csaf/provider-metadata.json`.

  - The content of the resource SHALL be equal to the resource accessible at `/.well-known/csaf/v2.0/provider-metadata.json`.
  - For file-based distribution servers, this MAY be achieved by using a symlink. Redirects SHALL NOT be used (cf. to requirement 7.1.9)

- Sometime before the roll-over-date, all existing CSAF 2.0 documents SHOULD be converted to CSAF 2.1.
- A the roll-over-date, a `provider-metadata.json` in conformance to CSAF 2.1 SHOULD be placed at `/.well-known/csaf/provider-metadata.json`.

  - The content of the resource SHALL be equal to the resource accessible at `/.well-known/csaf/v2.1/provider-metadata.json`.
  - For file-based distribution servers, this MAY be achieved by using a symlink. Redirects SHALL NOT be used (cf. to requirement 7.1.9)

- At the end of the transition period, the URL of the CSAF 2.0 `provider-metadata.json` SHOULD be removed from the `security.txt`.

  - The unmaintained CSAF 2.0 directory structure and files SHOULD be removed or made inaccessible.
  - The CSAF 2.0 documents MAY be archived.

If a DNS path (cf. section 7.1.10) is used instead of the well-known URL, the same process SHOULD be followed taking the rules below into account:

- The leading and authoritative `provider-metadata.json` is always served according to the DNS path requirement (see section 7.1.10).

  > This implies that a CSAF 2.0 `provider-metadata.json` is served at the beginning and replaced by the CSAF 2.1 `provider-metadata.json` at the roll-over-date.

- It is recommended to use the folders `v2.0` and `v2.1` to differentiate between the CSAF versions.

7.4.3 Archive of CSAF Document from Previous Version

The following rules apply for the archival of CSAF document from a previous version:

- This archive SHOULD be located in `/.well-known/csaf/archive/` and use the file name `v2.0.tar.zst`, `v2.0.tar.bz2` or `v2.0.tar.xz`.
- The CSAF documents within the archive MUST be sorted into folders according to requirement 11 in section 7.1.11 and be accompanied by a hash according to requirement 18 7.1.18.
- The archive MUST be accompanied by a hash of the same algorithm.
- Existing signatures MAY also be included into the archive. It is NOT RECOMMENDED to renew the signatures in the archive unless the archive is updated with new content.

7.4.4 Transition Process for a CSAF Aggregator

Similarly, to the transition process for a CSAF provider, the same process SHOULD be used for a CSAF aggregator. It is RECOMMENDED to use the following URLs during the process:

- `/.well-known/csaf-aggregator/aggregator.json` for the currently valid aggregator metadata.
- `/.well-known/csaf-aggregator/v2.0/aggregator.json` for a valid CSAF 2.0 `aggregator.json`
- `/.well-known/csaf-aggregator/v2.1/aggregator.json` for a valid CSAF 2.1 `aggregator.json`

A CSAF 2.1 aggregator MUST only sync and list CSAF 2.1 publishers and providers.

8 Safety, Security, and Data Protection Considerations

All safety, security, and data protection requirements relevant to the context in which CSAF documents are used MUST be translated into, and consistently enforced through, CSAF implementations and processes.

CSAF documents are based on JSON, thus the security considerations of [RFC8259] apply and are repeated here as service for the reader:

> Generally, there are security issues with scripting languages. JSON is a subset of JavaScript but excludes assignment and invocation.
> Since JSON's syntax is borrowed from JavaScript, it is possible to use that language's `eval()` function to parse most JSON texts (but not all; certain characters such as `U+2028 LINE SEPARATOR` and `U+2029 PARAGRAPH SEPARATOR` are legal in JSON but not JavaScript). This generally constitutes an unacceptable security risk, since the text could contain executable code along with data declarations. The same consideration applies to the use of eval()-like functions in any other programming language in which JSON texts conform to that language's syntax.

In addition, CSAF documents may be rendered by consumers in various human-readable formats like HTML or PDF. Thus, for security reasons, CSAF producers and consumers SHALL adhere to the following:

- CSAF producers SHOULD NOT emit messages that contain HTML, even though GitHub-flavoured Markdown is permitted. To include HTML, source code, or any other content that may be interpreted or executed by a CSAF consumer, e.g. to provide a proof-of-concept, the issuing party SHALL use Markdown's fenced code blocks or inline code option.
- Deeply nested markup can cause a stack overflow in the Markdown processor [GFMENG]. To reduce this risk, CSAF consumers SHALL use a Markdown processor that is hardened against such attacks. **Note**: One example is the GitHub fork of the `cmark` Markdown processor [GFMCMARK].
- To reduce the risk posed by possibly malicious CSAF files that do contain arbitrary HTML (including, for example, `data:image/svg+xml`), CSAF consumers SHALL either disable HTML processing (for example, by using the `--safe` option in the `cmark` Markdown processor) or run the resulting HTML through an HTML sanitizer.
- To reduce the risk posed by possibly malicious links within a CSAF document (including, for example, `javascript:` links), CSAF consumers SHALL either remove all actions from links (for example, by displaying them as standard text) or render only those actionable that are known to be safe (for example, determining that via the media type). CSAF consumers that are not prepared to deal with the security implications of formatted messages SHALL NOT attempt to render them and SHALL instead fall back to the corresponding plain text messages. As also any other programming code can be contained within a CSAF document, CSAF consumers SHALL ensure that none of the values of a CSAF document is run as code. Moreover, it SHALL be treated as unsafe (user) input.

  > Additional, supporting mitigation measures like retrieving only CSAF documents from trusted sources and check their integrity and signature before parsing the document SHOULD be in place to reduce the risk further.

The distribution requirements of CSAF data allow to specify domains as the value of the HTTP header `Access-Control-Allow-Origin`. While a wildcard (`*`) as header value usually prevents implementing browsers from sending credentials during the CORS request, the restriction to specified domains often enables sending credentials. Allowing several specified domains results in using dynamics on the server, which can widen the attack surface by using more code and configuration. Furthermore, this might reveal information about internal structures, e.g. which domains are allowed to send credentials, or which tools are used. Given that credentials from a browser are a potent tool in the event of an attack, restricting the origins seems to imply a higher risk and therefore less secure than allowing all domains without credentials.

As setting the `Access-Control-Allow-Origin` header potentially allows for cross site request forgery, it SHOULD only be served on files and directories containing CSAF data. For any restricted feeds, standard authentication methods SHOULD be used that are not send by web browsers if the wildcard is used as header value.

CSAF producers, CSAF consumers and CSAF validators SHOULD NOT automatically retrieve JSON schemas from a URL declared in CSAF documents as this poses a security risk. Loading files from an untrusted source can result in information

leakage or remotely triggered automated exploitation. If CSAF producers, CSAF consumers or CSAF validators provide a option to interactively or automatically load missing schemes, they SHALL point out the risks of setting the option and actively warn the user about it. Such option SHALL NOT be set by default. CSAF producers, CSAF consumers and CSAF validators SHOULD keep a local copy of all schemas necessary to fulfill their tasks. Tool developers SHOULD ensure to regularly check that those copies still reflect the current version of those schemas as they could have been altered, e.g. by an Errata to fix a bug. It is RECOMMENDED to do them at least before a release.

> Such checks can be automated, e.g. in the CI/CD pipeline.

9 Conformance

In the only subsection of this section, the conformance targets and clauses are listed. The clauses, matching the targets one to one, are listed in separate sub-subsections of the targets listing subsection.

The order in which targets, and their corresponding clauses appear is somewhat arbitrary as there is no natural order on such diverse roles participating in the document exchanging ecosystem.
Except for the target **CSAF Document**, all other 33 targets span a taxonomy of the complex CSAF ecosystems existing in and between diverse security advisory generating, sharing, and consuming communities.
In any case, there are no capabilities organized in increasing quality levels for targets because the security advisory sharing communities follow the chain link model. Instead, a single minimum capability level for every target is given to maintain important goals of providing a common framework for security advisories:

- Fast production, sharing, and actionable consumption of security advisories
- Consistent end to end automation through collaborating actors
- Clear baseline across the communities per this specification
- Additional per-community cooperative extensions which may flow back into future updates of this specification

9.1 Conformance Targets

This document defines requirements for the file format and for certain software components that interact with it. The entities (“conformance targets”) for which this document defines requirements are:

- **CSAF Document**: A security advisory text document in the format defined by this document.
- **CSAF Producer**: A program which emits output in the CSAF format.
- **CSAF Direct Producer**: An analysis tool which acts as a CSAF Producer.
- **CSAF Converter**: A CSAF Producer that transforms the output of an analysis tool from its native output format into the CSAF format.
- **CVRF CSAF Converter**: A CSAF Producer which takes a CVRF document as input and converts it into a valid CSAF Document.
- **CSAF Content Management System**: A program that is able to create, review and manage CSAF Documents and is able to preview their details as required by CSAF Viewer.
- **CSAF Post-Processor**: A CSAF Producer that transforms an existing CSAF Document into a new CSAF Document, for example, by removing or redacting elements according to sharing policies.
- **CSAF Modifier**: A CSAF Post-Processor which takes a CSAF Document as input and modifies the structure or values of properties. The output is a valid CSAF Document.
- **CSAF Translator**: A CSAF Post-Processor which takes a CSAF Document as input and translates values of properties into another language. The output is a valid CSAF Document.
- **CSAF Consumer**: A program that reads and interprets a CSAF Document.
- **CSAF Viewer**: A CSAF Consumer that reads a CSAF Document, displays a list of the results it contains, and allows an end user to view each result in the context of the artifact in which it occurs.
- **CSAF Management System**: A program that is able to manage CSAF Documents and is able to display their details as required by CSAF Viewer.
- **CSAF Asset Matching System**: A program that connects to or is an asset database and is able to manage CSAF Documents as required by CSAF Management System as well as matching them to assets of the asset database.
- **CSAF Basic Validator**: A program that reads a document and checks it against the JSON schema and performs mandatory tests.
- **CSAF Extended Validator**: A CSAF Basic Validator that additionally performs recommended tests.
- **CSAF Full Validator**: A CSAF Extended Validator that additionally performs informative tests.
- **CSAF SBOM Matching System**: A program that connects to or is an SBOM database and is able to manage CSAF Documents as required by CSAF Management System as well as matching them to SBOM components of the SBOM database.

- **CSAF 2.0 to CSAF 2.1 Converter**: A CSAF Producer which takes a CSAF 2.0 Document as input and converts it into a valid CSAF 2.1 Document.
- **CSAF Library**: A library that implements CSAF data capabilities.
- **CSAF Library with Basic Validation**: A CSAF Library that also satisfies the conformance target "CSAF Basic Validator".
- **CSAF Library with Extended Validation**: A CSAF Library that also satisfies the conformance target "CSAF Extended Validator".
- **CSAF Library with Full Validation**: A CSAF Library that also satisfies the conformance target "CSAF Full Validator".
- **CSAF Downloader**: A program that retrieves CSAF Documents in an automated fashion.
- **CSAF Withdrawer**: A CSAF Post-Processor that transforms a given CSAF into a Withdrawn one.
- **CSAF Superseder**: A CSAF Post-Processor that transforms a given CSAF into a Superseded one.
- **CSAF RVISC ID Updater**: A CSAF Post-Processor that updates vulnerability IDs in a given CSAF based on the entries in [RVISC].
- **CSAF Additional Test**: A test that is not yet defined in section 6.
- **CSAF Extension**: A specified JSON object conveying additional information different from the content that can be conveyed with the CSAF Core elements.
- **CSAF Extension Schema**: A JSON schema specifying the content and properties of a CSAF Extension.
- **CSAF Extension Overlay Test**: A test whose execution depends on the presence of the specifying CSAF Extension that extends or replaces a test standardized in this specification or a CSAF Additional Test.
- **CSAF Extension Additional Test**: A test whose execution depends on the presence of the specifying CSAF Extension that provides additional checks in the context of the CSAF Extension or the CSAF Document the extension is embedded in.
- **CSAF Extension Test**: A test that is either a CSAF Extension Overlay Test or a CSAF Extension Additional Test.
- **CSAF Extension Specification**: The specification of a single CSAF Extension and related material.
- **CSAF Extension Bundle**: A of compilation of machine-readable artifacts related to a single CSAF Extension.
- **CSAF Extension Package**: A of compilation of all artifacts related to a single CSAF Extension.
- **CSAF Extension Collection**: A set of multiple CSAF Extension Package.

9.1.1 Conformance Clause 1: CSAF Document

A text file or data stream satisfies the "CSAF Document" conformance profile if it:

- conforms to the syntax and semantics defined in section 2.2.
- conforms to the syntax and semantics defined in section 2.3.
- conforms to the syntax and semantics defined in section 2.4.
- conforms to the syntax and semantics defined in section 3.
- satisfies at least one profile defined in section 4.
- conforms to the syntax and semantics defined in section 5.
- does not fail any mandatory test defined in section 6.1 respectively its corresponding CSAF Extension Test.
- contains no extension that does not fulfill the conformance profile "CSAF Extension".
- contains no extension at any other path than the specified ones in sections 3.1.4.4, 3.2.2.14, 3.2.4.10.1, 3.2.4.17 and 3.2.5.

9.1.2 Conformance Clause 2: CSAF Producer

A program satisfies the "CSAF Producer" conformance profile if the program:

- produces output in the CSAF format, according to the conformance profile "CSAF Document".
- satisfies those normative requirements in section 3 and 8 that are designated as applying to CSAF Producers.

9.1.3 Conformance Clause 3: CSAF Direct Producer

An analysis tool satisfies the "CSAF Direct Producer" conformance profile if the analysis tool:

- satisfies the "CSAF Producer" conformance profile.
- additionally satisfies those normative requirements in section 3 that are designated as applying to "direct producers" or to "analysis tools".
- does not emit any objects, properties, or values which, according to section 3, are intended to be produced only by converters.

9.1.4 Conformance Clause 4: CSAF Converter

A converter satisfies the "CSAF Converter" conformance profile if the converter:

- satisfies the "CSAF Producer" conformance profile.
- additionally satisfies those normative requirements in section 3 that are designated as applying to converters.
- does not emit any objects, properties, or values which, according to section 3, are intended to be produced only by direct producers.

9.1.5 Conformance Clause 5: CVRF CSAF Converter

A program satisfies the "CVRF CSAF Converter" conformance profile if the program fulfills the following two groups of requirements:

Firstly, the program:

- satisfies the "CSAF Producer" conformance profile.
- takes only CVRF documents as input.
- outputs a warning that an additional property was detected and not converted if it detects an additional property in the input. Such a warning MUST include the additional property and its path. The CVRF CSAF Converter SHALL ignore that additional property during the conversion.
- includes in every warning the relevant paths and values from the original file that triggered the alert, unless otherwise specified in this standard.
- additionally satisfies the normative requirements given below.

Secondly, the program fulfills the following for all items of:

- value type `string` with format `date-time`: If the value contains a `60` in the seconds place, the CVRF CSAF Converter MUST replace the seconds and their fractions with `59.999999`. In addition, the converter outputs a warning that leap seconds are now prohibited in CSAF and the value has been replaced. The CVRF CSAF Converter SHOULD indicate in such warning message whether the value was a valid leap second or not.
- type `/$defs/branches_t`:
  - If any `prod:Branch` instance has the type `Legacy`, `Realm`, or `Resource`, the CVRF CSAF Converter MUST replace those with the category `product_name`. In addition, the converter outputs a warning that those types do not exist in CSAF 2.1 and have been replaced with the category `product_name`.

    > There is a chance, that this replacement is incorrect or another category is a better fit. Users of the converter are advised to check the content of such documents to make sure the conversion is correct or at least not misleading.

  - If any `Branch Type` appears multiple times along a path under `/prod:ProductTree/prod:Branch` and the `Branch Type` does not map to an excepted category according to test 6.1.57, the CVRF CSAF Converter MUST try to convert the data into a valid product tree by applying the following steps to the path:

1. If the stacked `Branch Type` is `Vendor`, the vendor items named `Open Source`, `NOASSERTION` `undefined` and `unknown` (white space, dash, hyphen, minus, underscore and case insensitive) MUST be removed.
2. If the stacked `Branch Type` is `Product Version` and the item directly before the first `Product Version` is a `Product Name`:
   * the category of the original `Product Name` item MUST be changed to `product_family` and
   * the category of the first `Product Version` item MUST be changed to `product_name` and
   * the value of the newly created `product_family` item MUST be prepended at the value of the newly created `product_name` item.

If the CVRF CSAF Converter is able to create a valid product tree, it MUST output a warning that an invalid product tree with stacked branch types was detected and resolved. Such a warning MUST include the invalid path as well as the branch types that were present multiple times.

> A tool MAY provide a non-default option to suppress this conversion step.

If the CVRF CSAF Converter is unable to create a valid product tree, it MUST output an error that an invalid product tree with stacked branch types was detected and could not be resolved. Such a error MUST include the invalid path as well as the branch types that were present multiple times.

> A tool MAY provide a non-default option to output the invalid document.

- type `/$defs/version_t`: If any element doesn't match the semantic versioning, replace the all elements of type `/$defs/version_t` with the corresponding integer version. For that, CVRF CSAF Converter sorts the items of `/document/tracking/revision_history` by `number` ascending according to the rules of CVRF. Then, it replaces the value of `number` with the index number in the array (starting with 1). The value of `/document/tracking/version` is replaced by value of `number` of the corresponding revision item. The match MUST be calculated by the original values used in the CVRF document. If this conversion was applied, for each Revision the original value of `cvrf:Number` MUST be set as `legacy_version` in the converted document.
- `/document/acknowledgments[]/organization` and `/vulnerabilities[]/acknowledgments[]/organization`: If more than one `cvrf:Organization` instance is given, the CVRF CSAF Converter converts the first one into the `organization`. In addition, the converter outputs a warning that information might be lost during conversion of document or vulnerability acknowledgment.
- `/document/category`:
  - If the `cvrf:DocumentType` is Security Advisory (case-insensitive), the CVRF CSAF Converter MUST try to convert the data into a valid CSAF Document in this profile according to CSAF 2.1.

    > A tool MAY offer rules to create the missing fixed products from version ranges, if applicable.

    If the CVRF CSAF Converter is unable to create a valid CSAF 2.1 Document according to the profile, it SHALL set the `category` value to `csaf_deprecated_security_advisory`.
  - If one or more CVRF elements containing an `xml:lang` attribute exist and their value is English or the document language of the CVRF document is unspecified, the following rules apply:
    * If the `cvrf:DocumentTitle` starts with the string `Superseded` or the `cvrf:DocumentType` starts with `Superseded` (case-insensitive), the CVRF CSAF Converter MUST try to convert all data into a valid CSAF Document in the profile "Superseded" according to CSAF 2.1.
    * If the `cvrf:DocumentTitle` starts with the string `Withdrawn` or the `cvrf:DocumentType` starts with `Withdrawn` (case-insensitive), the CVRF CSAF Converter MUST try to convert all data into a valid CSAF Document in the profile "Withdrawn" according to CSAF 2.1.

> A tool MAY provide a non-default option to remove or transform certain or all elements the hinder the creation of a valid CSAF Document according to the profile.

> A tool MAY support this detection for other languages.

If the CVRF CSAF Converter is unable to create a valid CSAF 2.1 Document according to the profile, it SHALL set the `category` according to the conversion rules and output a warning a potentially withdrawn CSAF Document was created which would result in an invalid CSAF.

- `/document/lang`: If one or more CVRF elements containing an `xml:lang` attribute exist and contain the exact same value, the CVRF CSAF Converter converts this value into `lang`. If the values of `xml:lang` attributes are not equal, the CVRF CSAF Converter outputs a warning that the language could not be determined and possibly a document with multiple languages was produced. In addition, it SHOULD also present all values of `xml:lang` attributes as a set in the warning.
- `/document/license_expression`: If any `cvrf:Note` item with `Type` `Legal Disclaimer` contains a valid SPDX license expression, the CVRF CSAF Converter SHALL convert this value into `license_expression`. In addition, the converter outputs an information that license expression was found and set as document license expression.
- `/document/notes`: If any `cvrf:Note` item contains one of the `category` and `title` combinations specified in 3.2.2.8, where the `title` is extended, the CVRF CSAF Converter SHALL try to identify whether that extension is a specific product name, version or family. In such case, the CVRF CSAF Converter SHALL try to add the corresponding products to the note item and output a warning that a potential product specific note has been discovered and products have been assigned to it. Such warning MUST also include the note and the assigned products. If the CVRF CSAF Converter is unable to create a valid object, it MUST remove the reference to the products and output a warning that a potential product specific note has been discovered and no products could been assigned to it.
- `/document/publisher/name` and `/document/publisher/namespace`: Sets the value as given in the configuration of the program or the corresponding argument the program was invoked with. If values from both sources are present, the program SHOULD prefer the latter one. The program SHALL NOT use hard-coded values.
- `/document/tracking/id`: If the element `cvrf:ID` contains any newline sequence or leading or trailing white space, the CVRF CSAF Converter removes those characters. In addition, the converter outputs a warning that the ID was changed.
- `/product_tree/product_path[]`: For each element `prod:Relationship`, the CVRF CSAF Converter MUST apply the following rules:
  - The value of the attribute `ProductReference` is set as the value of `beginning_product_reference`.
  - The first `prod:FullProductName` instance is converted into the `full_product_name`. If more than one `prod:FullProductName` instance is given, the CVRF CSAF Converter MUST output a warning that information might be lost during conversion of product paths. Such warning SHOULD contain the values of all `prod:FullProductName` instances skipped.
  - The CVRF CSAF Converter constructs the first item of `subpaths` by converting the value of `RelationType` into `category` and the value of the attribute `RelatesToProductReference` into `next_product_reference`.

    > A tool MAY provide an option to collapse chained product path elements into the appropriate number of product path elements, if this is possible without the loss of correct information. Usually, collapsing chained product paths is not possible if
    >
    > * a product (type: `full_product_name_t`) defined by a product path element in the chain is referenced in other parts of the document, or
    > * a product (type: `full_product_name_t`) defined by a product path element in the chain

contains a `product_identification_helper` element. For examples, see appendix 9.1.36.

- `/vulnerabilities[]/cwes[]`:
  - The CVRF CSAF Converter MUST remove all preceding and trailing white space from the `name`.
  - The CVRF CSAF Converter MUST determine the CWE specification version the given CWE was selected from by using the latest version that matches the `id` and `name` exactly and was published prior to the value of `/document/tracking/current_release_date` of the source document. If no such version exist, the first matching version published after the value of `/document/tracking/current_release_date` of the source document SHOULD be used.

    This is done to create a deterministic conversion.

    If the CWE does not match at all, the CVRF CSAF Converter MUST omit this CWE and output a warning that an invalid CWE was found and has been removed.
  - If a `vuln:CWE` instance refers to a CWE category or view, the CVRF CSAF Converter MUST omit this instance and output a warning that this CWE has been removed as its usage is not allowed in vulnerability mappings.
- `/vulnerabilities[]/disclosure_date`: If a `vuln:ReleaseDate` was given, the CVRF CSAF Converter MUST convert its value into the `disclosure_date` element.
- `/vulnerabilities[]/ids`: If a `vuln:ID` element is given, the CVRF CSAF Converter converts it into the first item of the `ids` array. Then, the CVRF CSAF Converter MUST execute the following steps at the converted object:
  - If an `system_name` was given, the CVRF CSAF Converter MUST test whether it belongs to a registered vulnerability ID system in RVISC.
  - If the `system_name` belongs to an RVISC entry, the CVRF CSAF Converter MUST execute test 6.2.53.
    - If the test passes, no further action is needed.
    - If the test fails, the CVRF CSAF Converter MUST try to convert the entry based on the mapping given in [RVISC-M].
      - If the mapping succeeds and passes test 6.2.53, the CVRF CSAF Converter MUST output a warning that an ID from a registered vulnerability system was detected and converted.
      - If the mapping succeeds but does not passes test 6.2.53 or the mapping fails, the CVRF CSAF Converter MUST output a warning that an ID from a registered vulnerability system was detected and but could not be converted automatically. Such warning MUST state the reason for failure. This includes also if the mapping is not implemented.

      A tool MAY provide a non-default option to suppress this conversion step.

      The output MUST include the original values and, if applicable, the converted ones.
  - If the `system_name` does not belong to an RVISC entry, the CVRF CSAF Converter MUST try to convert the entry based on the mapping given in [RVISC-M].
    - If no matching mapping exists, the CVRF CSAF Converter MUST output an information that an ID from a potentially unregistered vulnerability system was detected and no change occurred.
    - If the mapping succeeds and passes test 6.2.53, the CVRF CSAF Converter MUST output a warning that an ID from a vulnerability system with a known mapping was detected and converted.
    - If the mapping succeeds but does not passes test 6.2.53 or the mapping fails otherwise, the CVRF CSAF Converter MUST output a warning that an ID from a vulnerability system with a known mapping was detected and but could not be converted automatically. Such warning MUST state the reason for failure. This includes also if the mapping is not implemented.

    A tool MAY provide a non-default option to suppress this conversion step.

    The output MUST include the original values and, if applicable, the converted ones.

- `/vulnerabilities[]/metrics[]`:
  - For any CVSS v4 element, the CVRF CSAF Converter MUST compute the `baseSeverity` from the `baseScore` according to the rules of the applicable CVSS standard. (CSAF CVRF v1.2 predates CVSS v4.0.)
  - For any CVSS v3 element, the CVRF CSAF Converter MUST compute the `baseSeverity` from the `baseScore` according to the rules of the applicable CVSS standard.
  - If no `product_id` is given, the CVRF CSAF Converter appends all Product IDs which are listed under `../product_status` in the arrays `known_affected`, `first_affected` and `last_affected`. If none of these arrays exist, the CVRF CSAF Converter outputs an error that no matching Product ID was found for this score element.
  - If a `vectorString` is missing, the CVRF CSAF Converter outputs an error that the CVSS element could not be converted as the CVSS vector was missing. A CVRF CSAF Converter MAY offer a configuration option to delete such elements.
  - If there are CVSS v3.0 and CVSS v3.1 Vectors available for the same product, the CVRF CSAF Converter discards the CVSS v3.0 information and provide in CSAF only the CVSS v3.1 information.
  - To determine, which minor version of CVSS v3 is used and to evaluate a CVSS v4 that was wrongly inserted in a CVSS v3 element, the CVRF CSAF Converter uses the following steps:
    1. Retrieve the CVSS version from the CVSS vector, if present.
       Example 1:
       ```
       CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H => 3.1
       ```
    2. Retrieve the CVSS version from the CVSS element's namespace, if present. The CVRF CSAF Converter outputs a warning that this value was guessed from the element's namespace.
       Example 2:
       ```
       xmlns:cvssv31="https://www.first.org/cvss/cvss-v3.1.xsd"
       <!-- -->
       <cvssv31:ScoreSetV3>
       ```
       is handled the same as
       Example 3:
       ```
       <ScoreSetV3 xmlns="https://www.first.org/cvss/cvss-v3.1.xsd">
       ```
    3. Retrieve the CVSS version from the CVSS namespace given in the root element, if present. The CVRF CSAF Converter outputs a warning that this value was guessed from the global namespace. If more than one CVSS namespace is present and the element is not clearly defined via the namespace, this step MUST be skipped without a decision.
       Example 4:
       ```
       xmlns:cvssv3="https://www.first.org/cvss/cvss-v3.0.xsd" => 3.0
       ```
    4. Retrieve the CVSS version from a config value, which defaults to `3.0`. (As CSAF CVRF v1.2 predates CVSS v3.1.) The CVRF CSAF Converter outputs a warning that this value was taken from the config.

- `/vulnerabilities[]/metrics[]/content/cvss_v4`: If an external reference in the vulnerability linking to the official FIRST.org CVSS v4.0 calculator exists, the CVRF CSAF Converter MUST convert the vector given in the fragment into a `cvss_v4` object linked to all affected products of the vulnerability.

  > A tool MAY implement an option to suppress this conversion.

  If the CVRF CSAF Converter is unable to construct a valid object with the information given, the CVRF CSAF Converter SHALL remove the invalid `cvss_v4` object and output a warning that the automatic conversion of the CVSS v4.0 reference failed. Such warning SHOULD include the specific error that occurred.

- `/vulnerabilities[]/notes`: If any `vuln:Note` item contains one of the `category` and `title` combinations specified in 3.2.4.11, where the `title` is extended, the CVRF CSAF Converter SHALL try to identify whether that extension is a specific product name, version or family. In such case, the CVRF CSAF Converter SHALL try to add the corresponding products to the note item and output a warning that a potential product specific note has been discovered and products have been assigned to it. Such warning MUST also include the note and the assigned

products. If the CVRF CSAF Converter is unable to create a valid object, it MUST remove the reference to the products and output a warning that a potential product specific note has been discovered and no products could been assigned to it.

- `/vulnerabilities[]/remediations[]`:
  - If neither `product_ids` nor `group_ids` are given, the CVRF CSAF Converter appends all Product IDs which are listed under `../product_status` in the arrays `known_affected`, `first_affected` and `last_affected` into `product_ids`. If none of these arrays exist, the CVRF CSAF Converter outputs an error that no matching Product ID was found for this remediation element.
  - The CVRF CSAF Converter MUST convert any remediation with the type `Vendor Fix` into the category `optional_patch` if the product in question is in one of the product status groups "Not Affected" or "Fixed" for this vulnerability. Otherwise, the category `vendor_fix` MUST be set. If multiple products are associated with the remediation - either directly or through a product group - and the products belong to different product status groups, the CVRF CSAF Converter MUST duplicate the remediation, change the category in one instance to `optional_patch` and distribute the products accordingly as stated by the conversion rule.
  - The CVRF CSAF Converter MUST convert any remediation with the type `None Available` into the category `fix_planned` if the product in question is also listed in a remediation of the type `Vendor Fix` with a `Date` in the future or no `Date` at all. Consequently, the product MUST be removed from the remediation of the category `vendor_fix`. If it was the last product in that remediation, the remediation MUST be removed.
  - The CVRF CSAF Converter MUST remove any product from a remediation with the type `None Available` if the product in question is also listed in a remediation of the type `Vendor Fix` with a `Date` in the past or to the exact same time. If it was the last product in that remediation, the remediation MUST be removed.
  - In any other case, the CVRF CSAF Converter MUST preserve the product in the remediation of the category `none_available`.
  - The CVRF CSAF Converter MUST output a warning if a remediation was added, deleted or the value of the category was changed, including the products it was changed for.
- The CVRF CSAF Converter SHALL provide the JSON path where the warning occurred together with the warning.

9.1.6 Conformance Clause 6: CSAF Content Management System

A CSAF Content Management System satisfies the "CSAF Content Management System" conformance profile if the content management system:

- satisfies the "CSAF Producer" conformance profile.
- satisfies the "CSAF Viewer" conformance profile.
- provides at least the following management functions:
  - create new CSAF Documents
  - prefill CSAF Documents based on values given in the configuration (see below)
  - create a new version of an existing CSAF Document
  - checkout old versions of a CSAF Document
  - show all differences between versions of a CSAF Document
  - list all CSAF Documents within the system
  - delete CSAF Documents from the system
  - review CSAF Documents in the system
  - approve CSAF Documents
  - search for CSAF Documents by values of required fields at `document`-level or their children within the system

  - search for CSAF Documents by values of `cve` within the system
  - search for CSAF Documents based on properties of `product_tree`
  - filter on all properties which it is required to search for
  - export of CSAF Documents
  - show an audit log for each CSAF Document
  - identify the latest version of CSAF Documents with the same `/document/tracking/id`
  - suggest a `/document/tracking/id` based on the given configuration.
  - track of the version of CSAF Documents automatically and increment according to the versioning scheme (see also subsections of 3.1.13) selected in the configuration.
  - check that the document version is set correctly based on the changes in comparison to the previous version (see also subsections of 3.1.13).
  - suggest to use the document status `interim` if a CSAF Document is updated more frequent than the given threshold in the configuration (default: `3` weeks)
  - suggest to publish a new version of the CSAF Document with the document status `final` if the document status was `interim` and no new release has be done during the given threshold in the configuration (default: `6` weeks)

    > Note that the terms "publish", "publication" and their derived forms are used in this conformance profile independent of whether the specified target group is the public or a closed group.

  - support the following workflows:
    * "New Advisory": create a new advisory, request a review, provide review comments or approve it, resolve review comments; if the review approved it, the approval for publication can be requested; if granted the document status changes to `final` (or `interim` based on the selection in approval or configuration) and the advisory is provided for publication (manual or time-based)
    * "Update Advisory": open an existing advisory, create new revision & change content, request a review, provide review comments or approve it, resolve review comments; if the review approved it, the approval for publication can be requested; if granted the document status changes to `final` (or `interim` based on the selection in approval or configuration) and the advisory is provided for publication (manual or time-based)

- offers both: publication immediately or at a given date/time.
- automates handling of date/time and version.
- provides an API to retrieve all CSAF Documents which are currently in the status published.
- optionally provides an API to import or create new advisories from outside systems (e.g. bug tracker, CVD platform,...).
- provides a user management and support at least the following roles:
  - *Registered*: Able to see all published CSAF Documents (but only in the published version).
  - *Author*: inherits *Registered* permissions and also can Create and Edit Own (mostly used for automated creation, see above)
  - *Editor*: inherits *Author* permissions and can Edit (mostly used in PSIRT)
  - *Publisher*: inherits *Editor* permissions and can Change state and Review any (mostly used as HEAD of PSIRT or team lead)
  - *Reviewer*: inherits *Registered* permissions and can Review advisories assigned to him (might be a subject matter expert or management)
  - *Manager*: inherits *Publisher* permissions and can Delete; User management up to *Publisher*

  - *Administrator*: inherits *Manager* permissions and can Change the configuration

- may use groups to support client separation (multitenancy) and therefore restrict the roles to actions within their group. In this case, there MUST be a *Group configurator* which is able to change the values which are used to prefill fields in new advisories for that group. He might also do the user management for the group up to a configured level.
- prefills the following fields in new CSAF Documents with the values given below or based on the templates from configuration:
  - `/$schema` with the value prescribed by the schema
  - `/document/csaf_version` with the value prescribed by the schema
  - `/document/lang`
  - `/document/notes`
    * `legal_disclaimer` (Terms of use from the configuration)
    * `general` (General Security recommendations from the configuration)
  - `/document/tracking/current_release_date` with the current date
  - `/document/tracking/generator` and children
  - `/document/tracking/initial_release_date` with the current date
  - `/document/tracking/revision_history`
    * `date` with the current date
    * `number` (based on the templates according to the versioning scheme configured)
    * `summary` (based on the templates from configuration; default: "Initial version.")
  - `/document/tracking/status` with `draft`
  - `/document/tracking/version` with the value of `number` the latest `/document/tracking/revision_history[]` element
  - `/document/publisher` and children
  - `/document/category` (based on the templates from configuration)
- When updating an existing CSAF Document:
  - prefills all fields which have be present in the existing CSAF Document
  - adds a new item in `/document/tracking/revision_history[]`
  - updates the following fields with the values given below or based on the templates from configuration:
    * `/$schema` with the value prescribed by the schema
    * `/document/csaf_version` with the value prescribed by the schema
    * `/document/lang`
    * `/document/notes`
      · `legal_disclaimer` (Terms of use from the configuration)
      · `general` (General Security recommendations from the configuration)
    * `/document/tracking/current_release_date` with the current date
    * `/document/tracking/generator` and children
    * the new item in `/document/tracking/revision_history[]`
      · `date` with the current date
      · `number` (based on the templates according to the versioning scheme configured)
    * `/document/tracking/status` with `draft`
    * `/document/tracking/version` with the value of `number` the latest `/document/tracking/revision_history[]` element
    * `/document/publisher` and children

9.1.7 Conformance Clause 7: CSAF Post-Processor

A CSAF Post-Processor satisfies the "CSAF Post-Processor" conformance profile if the post-processor:

- satisfies the “CSAF Consumer” conformance profile.
- satisfies the “CSAF Producer” conformance profile.
- additionally satisfies those normative requirements in section 3 that are designated as applying to post-processors.

9.1.8 Conformance Clause 8: CSAF Modifier

A program satisfies the “CSAF Modifier” conformance profile if the program fulfills the two following groups of requirements:

The program:

- satisfies the “CSAF Post-Processor” conformance profile.
- adds, deletes or modifies at least one property, array, object or value of a property or item of an array.
- does not emit any objects, properties, or values which, according to section 9, are intended to be produced only by CSAF Translators.
- satisfies the normative requirements given below.

The resulting modified document:

- does not have the same `/document/tracking/id` as the original document. The modified document can use a completely new `/document/tracking/id` or compute one by appending the original `/document/tracking/id` as a suffix after an ID from the naming scheme of the issuer of the modified version. It SHOULD NOT use the original `/document/tracking/id` as a prefix.
- includes a reference to the original advisory as first element of the array `/document/references[]`.

9.1.9 Conformance Clause 9: CSAF Translator

A program satisfies the “CSAF Translator” conformance profile if the program fulfills the two following groups of requirements:

The program:

- satisfies the “CSAF Post-Processor” conformance profile.
- translates at least one value.
- preserves the same semantics and form across translations.
- satisfies the normative requirements given below and does not add or remove other elements than required below.

The resulting translated document:

- does not use the same `/document/tracking/id` as the original document. The translated document can use a completely new `/document/tracking/id` or compute one by using the original `/document/tracking/id` as a prefix and adding an ID from the naming scheme of the issuer of the translated version. It SHOULD NOT use the original `/document/tracking/id` as a suffix. If an issuer uses a CSAF Translator to publish his advisories in multiple languages they MAY use the combination of the original `/document/tracking/id` and translated `/document/lang` as a `/document/tracking/id` for the translated document.

  > Note that the term “publish” is used in this conformance profile independent of whether the specified target group is the public or a closed group.

- provides the `/document/lang` property with a value matching the language of the translation.
- provides the `/document/source_lang` to contain the language of the original document (and SHOULD only be set by CSAF Translators).
- has the value `translator` set in `/document/publisher/category`
- includes a reference to the original advisory as first element of the array `/document/references[]`.

- MAY contain translations for elements in arrays of `references_t` after the first element. However, it MUST keep the original URLs as references at the end.

9.1.10 Conformance Clause 10: CSAF Consumer

A processor satisfies the "CSAF Consumer" conformance profile if the processor:

- reads CSAF Documents and interprets them according to the semantics defined in section 3 and 5.
- satisfies those normative requirements in section 2.4, 3, 5 and 8 that are designated as applying to CSAF Consumers.

9.1.11 Conformance Clause 11: CSAF Viewer

A viewer satisfies the "CSAF Viewer" conformance profile if the viewer fulfills the two following groups of requirements:

The viewer:

- satisfies the "CSAF Consumer" conformance profile.
- satisfies those normative requirements in section 3, 5 and 8 that are designated as applying to CSAF Viewers.
- satisfies the normative requirements given below.

For each CVSS-Score in `/vulnerabilities[]/metrics[]` the viewer:

- preferably shows the `vector` if there is an inconsistency between the `vector` and any other sibling attribute.
- SHOULD prefer the item of `metrics[]` for each `product_id` which originates from the document author (and therefore has no property `source`) and has the highest CVSS Base Score and newest CVSS version (in that order) if a `product_id` is listed in more than one item of `metrics[]`.

9.1.12 Conformance Clause 12: CSAF Management System

A CSAF Management System satisfies the "CSAF Management System" conformance profile if the management system:

- satisfies the "CSAF Viewer" conformance profile.
- provides at least the following management functions:
  - add new CSAF Documents (e.g. from file system or URL) to the system
  - list all CSAF Documents within the system
  - delete CSAF Documents from the system
  - comment on CSAF Documents in the system
  - mark CSAF Documents as read in the system
  - search for CSAF Documents by values of required fields at `document`-level or their children within the system
  - search for CSAF Documents by values of `cve` within the system
  - search for CSAF Documents based on properties of `/product_tree`
  - filter on all properties which it is required to search for
  - sort on all properties which it is required to search for
  - sort on CVSS scores and `/document/aggregate_severity/text`
- identifies the latest version of CSAF Documents with the same `/document/tracking/id`.
- is able to show the difference between 2 versions of a CSAF Document with the same `/document/tracking/id`.

9.1.13 Conformance Clause 13: CSAF Asset Matching System

A CSAF Asset Matching System satisfies the "CSAF Asset Matching System" conformance profile if the asset matching system:

- satisfies the "CSAF Management System" conformance profile.
- is an asset database or connects to one.
- matches the CSAF Documents within the system to the respective assets. This might be done with a probability which gives the end user the chance to broaden or narrow the results. The process of matching is also referred to as "run of the asset matching module".
- provides for each product of the asset database a list of matched advisories.
- provides for each asset of the asset database a list of matched advisories.
- provides for each CSAF Document a list of matched product of the asset database.
- provides for each CSAF Document a list of matched asset of the asset database.
- provides for each vulnerability within a CSAF Document the option to mark a matched asset in the asset database as "not remediated", "remediation in progress", or "remediation done". A switch to mark all assets at once MAY be implemented.
- does not bring up a newer revision of a CSAF Document as a new match if the remediation for the matched product or asset has not changed.
- detects the usage semantic version (as described in section 3.1.13.2).
- is able to trigger a run of the asset matching module:
  - manually:
    - per CSAF Document
    - per list of CSAF Documents
    - per asset
    - per list of assets
  - automatically:
    - when a new CSAF Document is inserted (for this CSAF Document)
    - when a new asset is inserted (for this asset)
    - when the Major version in a CSAF Document with semantic versioning changes (for this CSAF Document)

    > These also apply if more than one CSAF Document or asset was added. To reduce the computational efforts the runs can be pooled into one run which fulfills all the tasks at once (batch mode).

  - Manually and automatically triggered runs SHOULD NOT be pooled.
- provides at least the following statistics for the count of assets:
  - matching that CSAF Document at all
  - marked with a given status

9.1.14 Conformance Clause 14: CSAF Basic Validator

A program satisfies the "CSAF Basic Validator" conformance profile if the program:

- reads documents and performs a check against the JSON schema, including also the format validation (cf. section 2.2).
- performs all tests of the preset `mandatory` as given in section 6.4.1.
- does not change the CSAF Documents.
- satisfies those normative requirements in sections 2.4, 3 6.1, 6.4, and 8 that are designated as applying to CSAF Validators.
- outputs a warning if an "not implemented warning" occurs as the validation status might not be correct.

A CSAF Basic Validator MAY provide one or more additional functions:

- Only run one or more selected mandatory tests.

- Apply quick fixes as specified in the standard.
- Apply additional quick fixes as implemented by the vendor.

A CSAF Basic Validator MAY implement CSAF Additional Tests. In that case, it MUST make through its documentation available which tests are implemented.

9.1.15 Conformance Clause 15: CSAF Extended Validator

A CSAF Basic Validator satisfies the "CSAF Extended Validator" conformance profile if the CSAF Basic Validator:

- satisfies the "CSAF Basic Validator" conformance profile.
- additionally performs all tests of the preset `recommended` as given in section 6.4.1.
- additionally satisfies those normative requirements in section 6.2 that are designated as applying to CSAF Validators.

A CSAF Extended Validator MAY provide an additional function to only run one or more selected recommended tests.

9.1.16 Conformance Clause 16: CSAF Full Validator

A CSAF Extended Validator satisfies the "CSAF Full Validator" conformance profile if the CSAF Extended Validator:

- satisfies the "CSAF Extended Validator" conformance profile.
- additionally performs all tests of the preset `informative` as given in section 6.4.1.
- provides an option to additionally use a custom dictionary for test 6.3.8.
- additionally satisfies those normative requirements in section 6.3 that are designated as applying to CSAF Validators.

A CSAF Full Validator MAY provide an additional function to only run one or more selected informative tests.

9.1.17 Conformance Clause 17: CSAF SBOM Matching System

A CSAF SBOM Matching System satisfies the "CSAF SBOM Matching System" conformance profile if the SBOM matching system:

- satisfies the "CSAF Management System" conformance profile.
- is an SBOM database or connects to one.

  > A repository or any other location that can be queried for SBOMs and their content is also considered an SBOM database.

- matches the CSAF Documents within the system to the respective SBOM components. This might be done with a probability which gives the user the chance to broaden or narrow the results. The process of matching is also referred to as "run of the SBOM matching module".
- provides for each SBOM of the SBOM database a list of matched advisories.
- provides for each SBOM component of the SBOM database a list of matched advisories.
- provides for each CSAF Document a list of matched SBOMs of the SBOM database.
- provides for each CSAF Document a list of matched SBOM components of the SBOM database.
- provides for each vulnerability within a CSAF Document the option to mark a matched SBOM component in the SBOM database as "not remediated", "remediation in progress", or "remediation done". A switch to mark all SBOM component at once MAY be implemented.
- does not bring up a newer revision of a CSAF Document as a new match if the remediation for the matched SBOM or SBOM component has not changed.
- detects the usage semantic version (as described in section 3.1.13.2).

- is able to trigger a run of the SBOM matching module:
  - manually:
    - per CSAF Document
    - per list of CSAF Documents
    - per SBOM component
    - per list of SBOM components
  - automatically:
    - when a new CSAF Document is inserted (for this CSAF Document)
    - when a new SBOM component is inserted (for this SBOM component)
    - when the Major version in a CSAF Document with semantic versioning changes (for this CSAF Document)

    > These also apply if more than one CSAF Document or SBOM component was added. To reduce the computational efforts the runs can be pooled into one run which fulfills all the tasks at once (batch mode).

  > Manually and automatically triggered runs should not be pooled.

- provides at least the following statistics for the count of SBOM component:
  - matching that CSAF Document at all
  - marked with a given status

9.1.18 Conformance Clause 18: CSAF 2.0 to CSAF 2.1 Converter

A program satisfies the "CSAF 2.0 to CSAF 2.1 Converter" conformance profile if the program fulfills the following two groups of requirements:

Firstly, the program:

- satisfies the "CSAF Producer" conformance profile.
- takes only CSAF 2.0 Documents as input.
- outputs a warning that an additional property was detected and not converted if it detects an additional property in the input. Such a warning MUST include the additional property and its path. The CSAF 2.0 to CSAF 2.1 Converter SHALL ignore that additional property during the conversion.
- includes in every warning the relevant paths and values from the original file that triggered the alert, unless otherwise specified in this standard.
- additionally satisfies the normative requirements given below.

Secondly, the program fulfills the following for all items of:

- value type `string` with format `date-time`: If the value contains a `60` in the seconds place, the CSAF 2.0 to CSAF 2.1 Converter MUST replace the seconds and their fractions with `59.999999`. In addition, the converter outputs a warning that leap seconds are now prohibited in CSAF and the value has been replaced. The CSAF 2.0 to CSAF 2.1 Converter SHOULD indicate in such warning message whether the value was a valid leap second or not.
- type `/$defs/branches_t`:
  - If a branch item uses the category `legacy`, the CSAF 2.0 to CSAF 2.1 Converter MUST replace it with the category `product_name`. In addition, the converter outputs a warning that this type does not exist in CSAF 2.1 and have been replaced with the category `product_name`.

    > There is a chance, that this replacement is incorrect or another category is a better fit. Users of the converter are advised to check the content of such documents to make sure the conversion is correct or at least not misleading.

- If any branch category appears multiple times along a path under `/product_tree/branches` and the category is not an excepted one according to test 6.1.57, the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert the data into a valid product tree by applying the following steps to the path:

  1. If the stacked branch category is `vendor`, the vendor items named `Open Source`, `NOASSERTION`, `undefined` and `unknown` (white space, dash, hyphen, minus, underscore and case insensitive) MUST be removed.
  2. If the stacked branch category is `product_version` and the item directly before the first `product_version` is a `product_name`:
     * the category of the original `product_name` item MUST be changed to `product_family` and
     * the category of the first `product_version` item MUST be changed to `product_name` and
     * the value of the newly created `product_family` item MUST be prepended at the value of the newly created `product_name` item.

  If the CSAF 2.0 to CSAF 2.1 Converter is able to create a valid product tree, it MUST output a warning that an invalid product tree with stacked branch categories was detected and resolved. Such a warning MUST include the invalid path as well as the branch categories that were present multiple times.

  > A tool MAY provide a non-default option to suppress this conversion step.

  If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid product tree, it MUST output an error that an invalid product tree with stacked branch categories was detected and could not be resolved. Such a error MUST include the invalid path as well as the branch categories that were present multiple times.

  > A tool MAY provide a non-default option to output the invalid document.

- If the branch categories `product_version` and `product_version_range` appear along a path under `/product_tree/branches`, the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert the data into a valid product tree by applying the following steps to the path:

  1. If the branch category `product_version` occurs before the `product_version_range` and the item directly before the first `product_version` is a `product_name`:
     * the category of the original `product_name` item MUST be changed to `product_family` and
     * the category of the first `product_version` item MUST be changed to `product_name` and
     * the value of the newly created `product_family` item MUST be prepended at the value of the newly created `product_name` item.
  2. If the branch category `product_version` occurs before the `product_version_range` and the item directly before the first `product_version` is a `product_family`:
     * the category of the first `product_version` item MUST be changed to `product_name` and
     * the value of the direct ancestor `product_family` item MUST be prepended at the value of the newly created `product_name` item.

  If the CSAF 2.0 to CSAF 2.1 Converter is able to create a valid product tree, it MUST output a warning that an invalid product tree with branch categories `product_version` and `product_version_range` in one path was detected and resolved. Such a warning MUST include the invalid path as well as the branch category items changed.

  > A tool MAY provide a non-default option to suppress this conversion step.

  If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid product tree, it MUST output an error that an invalid product tree with branch categories `product_version` and `product_version_range` in one path was detected and could not be resolved. Such a error MUST include the invalid path as well as the branch category items.

  > A tool MAY provide a non-default option to output the invalid document.

- If the value of `name` of an item categorized as `product_version_range` and contains an upper open ended product version range, the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert the data into a valid product tree by applying the following steps to the path:

  1. If value of `name` consists only of one version constraint:
     * the category of the original `product_version_range` item MUST be changed to `product_version` and
     * the version MUST be extracted from the original value and set as new value of `name`.
  2. If value of `name` consists of more than one version constraint and the upper open range is the last version constraint:
     * the category of the original `product_version_range` item MUST be kept and
     * the value `name` MUST be converted into the product version range ending with the version the upper open ended product version if that does not change the inclusion boundaries.

  If the CSAF 2.0 to CSAF 2.1 Converter is able to create a valid product tree, it MUST output a warning that an invalid product tree with an upper open ended product version range in one path was detected and resolved. Such a warning MUST include the invalid path as well as the original and new value.

  > A tool MAY provide a non-default option to suppress this conversion step.

  If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid product tree, it MUST output an error that an invalid product tree with an upper open ended product version range in one path was detected and could not be resolved. Such a error MUST include the invalid path as well as the original and new value.

  > A tool MAY provide a non-default option to output the invalid document.

- If the value of `name` of an item categorized as `product_version_range` contains just a single version, the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert the data into a valid product tree by applying the following steps to the path:

  1. If value of `name` is in the vers format:
     * the category of the original `product_version_range` item MUST be changed to `product_version` and
     * the version MUST be extracted from the version constraint and set as new value of `name`.
  2. If value of `name` is in the vls format:
     * the category of the original `product_version_range` item MUST be changed to `product_version` and
     * the value `name` MUST be kept unchanged.

  If the CSAF 2.0 to CSAF 2.1 Converter is able to create a valid product tree, it MUST output a warning that an invalid product tree with a `product_version` declared as `product_version_range` in one path was detected and resolved. Such a warning MUST include the invalid path as well as value of the product version.

  > A tool MAY provide a non-default option to suppress this conversion step.

  If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid product tree, it MUST output an error that an invalid product tree with a `product_version` declared as `product_version_range` in one path was detected and could not be resolved. Such a error MUST include the invalid path as well as value of the product version.

  > A tool MAY provide a non-default option to output the invalid document.

- type `/$defs/full_product_name_t/product_identification_helper/cpe`: If a CPE is invalid, the CSAF 2.0 to CSAF 2.1 Converter SHOULD removed the invalid value and output a warning that an invalid CPE was detected and removed. Such a warning MUST include the invalid CPE.

- type `/$defs/full_product_name_t/product_identification_helper/hashes[]/file_hashes[]`
  If the algorithm is known to the implementation or mentioned in this standard, the CSAF 2.0 to CSAF 2.1 Converter MUST ensure its spelling is exactly as prescribed by this standard. If the algorithm is unknown to the implementation, the CSAF 2.0 to CSAF 2.1 Converter MUST convert it to lowercase and output a warning that an unknown hash algorithm was detected and converted. Such a warning MUST include the invalid path as well as value of the algorithm.

  A tool MAY provide a non-default option to suppress this conversion step.

- type `/$defs/full_product_name_t/product_identification_helper/hashes[]/file_hashes[]`
  The CSAF 2.0 to CSAF 2.1 Converter MUST convert the value into a lowercase string.

- type `/$defs/full_product_name_t/product_identification_helper/model_numbers[]`:

  - If a model number is given that does not end on a star, the CSAF 2.0 to CSAF 2.1 Converter SHOULD add a `*` to the end and output a warning that a partial model number was detected and a star has been added. Such a warning MUST include the model number.
  - If the model number contains a `\`, the CSAF 2.0 to CSAF 2.1 Converter MUST escape it by inserting an additional `\` before the character.
  - If the model number contains multiple unescaped `*` after the conversion, the CSAF 2.0 to CSAF 2.1 Converter MUST remove the entry and output a warning that a model number with multiple stars was detected and removed. Such a warning MUST include the model number.

  A tool MAY provide a non-default option to interpret all model numbers as complete and therefore does not add any stars.

  A tool MAY provide a non-default option to interpret the `?` in all model numbers as part of the model number itself and therefore escape it.

  A tool MAY provide a non-default option to interpret the `*` in all model numbers as part of the model number itself and therefore escape it.

- type `/$defs/full_product_name_t/product_identification_helper/purls`: If a `/$defs/full_pr` is given, the CSAF 2.0 to CSAF 2.1 Converter MUST convert it into the first item of the corresponding `purls` array.

- type `/$defs/full_product_name_t/product_identification_helper/serial_numbers[]`:

  - If a serial number is given that does not end on a star, the CSAF 2.0 to CSAF 2.1 Converter SHOULD add a `*` to the end and output a warning that a partial serial number was detected and a star has been added. Such a warning MUST include the serial number.
  - If the serial number contains a `\`, the CSAF 2.0 to CSAF 2.1 Converter MUST escape it by inserting an additional `\` before the character.
  - If the serial number contains multiple unescaped `*` after the conversion, the CSAF 2.0 to CSAF 2.1 Converter MUST remove the entry and output a warning that a serial number with multiple stars was detected and removed. Such a warning MUST include the serial number.

  A tool MAY provide a non-default option to interpret all serial numbers as complete and therefore does not add any stars.

  A tool MAY provide a non-default option to interpret the `?` in all serial numbers as part of the serial number itself and therefore escape it.

  A tool MAY provide a non-default option to interpret the `*` in all serial numbers as part of the serial number itself and therefore escape it.

- type `/$defs/full_product_name_t/product_identification_helper/skus[]`:

  - If a stock keeping unit is given that does not end on a star, the CSAF 2.0 to CSAF 2.1 Converter SHOULD add a `*` to the end and output a warning that a partial stock keeping unit was detected and a star has been added. Such a warning MUST include the stock keeping unit.
  - If the stock keeping unit contains a `\`, the CSAF 2.0 to CSAF 2.1 Converter MUST escape it by inserting an additional `\` before the character.
  - If the stock keeping unit contains multiple unescaped `*` after the conversion, the CSAF 2.0 to CSAF 2.1 Converter MUST remove the entry and output a warning that a stock keeping unit with multiple stars was detected and removed. Such a warning MUST include the stock keeping unit.

> A tool MAY provide a non-default option to interpret all stock keeping units as complete and therefore does not add any stars.

> A tool MAY provide a non-default option to interpret the `?` in all stock keeping units as part of the stock keeping unit itself and therefore escape it.

> A tool MAY provide a non-default option to interpret the `*` in all stock keeping units as part of the stock keeping unit itself and therefore escape it.

- `/$schema`: The CSAF 2.0 to CSAF 2.1 Converter MUST set property with the value prescribed by the schema.
- `/document/category`:
  - If the `category` equals `csaf_security_advisory`, the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert the data into a valid CSAF Document in this profile according to CSAF 2.1. For any version range of affected products that uses the strict `<`, i.e. not `<=`, as comparator of the last version constraint, the CSAF 2.0 to CSAF 2.1 Converter SHOULD add a new product with the version of the last constraint and add that in the appropriate places as `fixed`. The CSAF 2.0 to CSAF 2.1 Converter MUST output a warning that a product was added to the `product_tree` and the corresponding `/vulnerabilities[]`. Such warning MUST contain the full product name and its path as well as the paths of the `/vulnerabilities[]` it was added to. If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid CSAF 2.1 Document according to the profile, it SHALL set the `category` value to `csaf_deprecated_security_advisory`.
  - If the `/document/lang` is English or unspecified, the following rules apply:
    * If the `/document/title` starts with the string `Superseded` or the `/document/category` has the value `Superseded` (case-insensitive), the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert all data into a valid CSAF Document in the profile “Superseded” according to CSAF 2.1.
    * If the `/document/title` starts with the string `Withdrawn` or the `/document/category` has the value `Withdrawn` (case-insensitive), the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert all data into a valid CSAF Document in the profile “Withdrawn” according to CSAF 2.1.

    > A tool MAY provide a non-default option to remove or transform certain or all elements the hinder the creation of a valid CSAF Document according to the profile.

    > A tool MAY support this detection for other languages.

    If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid CSAF 2.1 Document according to the profile, it SHALL set the `category` of the original CSAF Document and output a warning a potentially withdrawn CSAF Document was created which would result in an invalid CSAF.

- `/document/csaf_version`: The CSAF 2.0 to CSAF 2.1 Converter MUST update the value to `2.1`.
- `/document/distribution/tlp/label`: If a TLP label is given, the CSAF 2.0 to CSAF 2.1 Converter MUST convert it according to the table below:

CSAF 2.0 (using TLP v1.0)	CSAF 2.1 (using TLP v2.0)
`TLP:WHITE`	`TLP:CLEAR`

CSAF 2.0 (using TLP v1.0)	CSAF 2.1 (using TLP v2.0)
`TLP:GREEN`	`TLP:GREEN`
`TLP:AMBER`	`TLP:AMBER`
`TLP:RED`	`TLP:RED`

If `/document/distribution/text` contains the string `TLP v2.0: TLP:<ValidTLPLabel>`, the CSAF 2.0 to CSAF 2.1 Converter SHOULD provide an option to use this label instead. If the TLP label changes through such conversion in a way that is not reflected in the table above, the the CSAF 2.0 to CSAF 2.1 Converter MUST output a warning that the TLP label was taken from the distribution text. Such a warning MUST include both values: the converted one based on the table and the one from the distribution text.

> This is a common case for CSAF 2.0 Documents labeled as `TLP:RED` but actually intended to be `TLP:AMBER+STRICT`.

If no TLP label was given, the CSAF 2.0 to CSAF 2.1 Converter SHOULD assign `TLP:CLEAR` and output a warning that the default TLP has been set.

- `/document/license_expression`: If any `/document/notes` item in with `category` `legal_disclaimer` contains a valid SPDX license expression, the CSAF 2.0 to CSAF 2.1 Converter SHALL convert this value into `license_expression`. In addition, the converter outputs an information that license expression was found and set as document license expression.
- `/document/notes`: If any `/document/notes` item contains one of the `category` and `title` combinations specified in 3.2.2.8, where the `title` is extended, the CSAF 2.0 to CSAF 2.1 Converter SHALL try to identify whether that extension is a specific product name, version or family. In such case, the CSAF 2.0 to CSAF 2.1 Converter SHALL try to add the corresponding products to the note item and output a warning that a potential product specific note has been discovered and products have been assigned to it. Such warning MUST also include the note and the assigned products. If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid object, it MUST remove the reference to the products and output a warning that a potential product specific note has been discovered and no products could been assigned to it.
- `/document/publisher/category`: If the value is `other`, the CSAF 2.0 to CSAF 2.1 Converter SHOULD output a warning that some parties have been regrouped into the new value `multiplier`. An option to suppress this warning MUST exist. In addition, an option SHOULD be provided to set the value to `multiplier`.
- `/document/title`: If the value contains the `/document/tracking/id`, the CSAF 2.0 to CSAF 2.1 Converter MUST remove the `/document/tracking/id` from the `/document/title`. In addition, separating characters including but not limited to white space, colon, dash and brackets MUST be removed.
- `/product_tree/product_path[]`: For each element in `/product_tree/relationships[]`, the CSAF 2.0 to CSAF 2.1 Converter MUST apply the following rules:
  - The value of `product_reference` is set as the value of `beginning_product_reference`.
  - The CSAF 2.0 to CSAF 2.1 Converter constructs the first item of `subpaths` by setting the value of `category` into `category` and the value of `relates_to_product_reference` into `next_product_reference`.

    > A tool MAY provide an option to collapse chained product path elements into the appropriate number of product path elements, if this is possible without the loss of correct information. Usually, collapsing chained product paths is not possible if
    >
    > * a product (type: `full_product_name_t`) defined by a product path element in the chain is referenced in other parts of the document, or
    > * a product (type: `full_product_name_t`) defined by a product path element in the chain contains a `product_identification_helper` element. For examples, see appendix 9.1.36.

- `/vulnerabilities[]/cwes[]`:
  - The CSAF 2.0 to CSAF 2.1 Converter MUST remove all preceding and trailing white space from the `name`.
  - The CSAF 2.0 to CSAF 2.1 Converter MUST determine the CWE specification version the given CWE was selected from by using the latest version that matches the `id` and `name` exactly and was published prior to the value of `/document/tracking/current_release_date` of the source document. If no such version exist, the first matching version published after the value of `/document/tracking/current_release_date` of the source document SHOULD be used.

    This is done to create a deterministic conversion.

    The tool SHOULD implement an option to use the latest available CWE version at the time of the conversion that still matches.
- `/vulnerabilities[]/disclosure_date`: If a `release_date` was given, the CSAF 2.0 to CSAF 2.1 Converter MUST convert its value as value into the `disclosure_date` element.
- `/vulnerabilities[]/ids`: If an `system_name` was given, the CSAF 2.0 to CSAF 2.1 Converter MUST test whether it belongs to a registered vulnerability ID system in RVISC.
  - If the `system_name` belongs to an RVISC entry, the CSAF 2.0 to CSAF 2.1 Converter MUST execute test 6.2.53.
    * If the test passes, no further action is needed.
    * If the test fails, the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert the entry based on the mapping given in [RVISC-M].
      · If the mapping succeeds and passes test 6.2.53, the CSAF 2.0 to CSAF 2.1 Converter MUST output a warning that an ID from a registered vulnerability system was detected and converted.
      · If the mapping succeeds but does not passes test 6.2.53 or the mapping fails, the CSAF 2.0 to CSAF 2.1 Converter MUST output a warning that an ID from a registered vulnerability system was detected and but could not be converted automatically. Such warning MUST state the reason for failure. This includes also if the mapping is not implemented.

    A tool MAY provide a non-default option to suppress this conversion step.

    The output MUST include the original values and, if applicable, the converted ones.
  - If the `system_name` does not belong to an RVISC entry, the CSAF 2.0 to CSAF 2.1 Converter MUST try to convert the entry based on the mapping given in [RVISC-M].
    * If no matching mapping exists, the CSAF 2.0 to CSAF 2.1 Converter MUST output an information that an ID from a potentially unregistered vulnerability system was detected and no change occurred.
    * If the mapping succeeds and passes test 6.2.53, the CSAF 2.0 to CSAF 2.1 Converter MUST output a warning that an ID from a vulnerability system with a known mapping was detected and converted.
    * If the mapping succeeds but does not passes test 6.2.53 or the mapping fails otherwise, the CSAF 2.0 to CSAF 2.1 Converter MUST output a warning that an ID from a vulnerability system with a known mapping was detected and but could not be converted automatically. Such warning MUST state the reason for failure. This includes also if the mapping is not implemented.

    A tool MAY provide a non-default option to suppress this conversion step.

    The output MUST include the original values and, if applicable, the converted ones.
- `/vulnerabilities[]/metrics[]/content/cvss_v4`: If an external reference in the vulnerability linking to the official FIRST.org CVSS v4.0 calculator exists, the CSAF 2.0 to CSAF 2.1 Converter MUST convert the vector given in the fragment into a `cvss_v4` object linked to all affected products of the vulnerability.

A tool MAY implement an option to suppress this conversion.

If the CSAF 2.0 to CSAF 2.1 Converter is unable to construct a valid object with the information given, the CSAF 2.0 to CSAF 2.1 Converter SHALL remove the invalid `cvss_v4` object and output a warning that the automatic conversion of the CVSS v4.0 reference failed. Such warning SHOULD include the specific error that occurred.

- `/vulnerabilities[]/metrics[]/content/ssvc_v2`: If a SSVC vector or decision points of an SSVC vector are given in an item of `notes` of the current vulnerability using the `title` `SSVC` and the `category` `other`, the CSAF 2.0 to CSAF 2.1 Converter MUST convert that data into the `ssvc_v2` object within the current vulnerability. If the CSAF 2.0 to CSAF 2.1 Converter is able to construct a valid object without loosing any information, the corresponding `notes` item SHALL be removed. If the CSAF 2.0 to CSAF 2.1 Converter is unable to construct a valid object with the information given, the CSAF 2.0 to CSAF 2.1 Converter SHALL remove the invalid `ssvc_v2` object, keep the original item of `notes` and output a warning that the automatic conversion of the SSVC data failed. If the CSAF 2.0 to CSAF 2.1 Converter would loose information during the conversion, the CSAF 2.0 to CSAF 2.1 Converter SHALL remove the `ssvc_v2` object, keep the original item of `notes` and output a warning that the automatic conversion of the SSVC data would lead to loosing information.
- `/vulnerabilities[]/notes`: If any `/vulnerabilities[]/notes` item contains one of the `category` and `title` combinations specified in 3.2.4.11, where the `title` is extended, the CSAF 2.0 to CSAF 2.1 Converter SHALL try to identify whether that extension is a specific product name, version or family. In such case, the CSAF 2.0 to CSAF 2.1 Converter SHALL try to add the corresponding products to the note item and output a warning that a potential product specific note has been discovered and products have been assigned to it. Such warning MUST also include the note and the assigned products. If the CSAF 2.0 to CSAF 2.1 Converter is unable to create a valid object, it MUST remove the reference to the products and output a warning that a potential product specific note has been discovered and no products could been assigned to it.
- `/vulnerabilities[]/remediations[]`:
  - The CSAF 2.0 to CSAF 2.1 Converter MUST convert any remediation with the category `vendor_fix` into the category `optional_patch` if the product in question is in one of the product status groups “Not Affected” or “Fixed” for this vulnerability. Otherwise, the category `vendor_fix` MUST stay the same. If multiple products are associated with the remediation - either directly or through a product group - and the products belong to different product status groups, the CSAF 2.0 to CSAF 2.1 Converter MUST duplicate the remediation, change the category in one instance to `optional_patch` and distribute the products accordingly as stated by the conversion rule.
  - The CSAF 2.0 to CSAF 2.1 Converter MUST convert any remediation with the category `none_available` into the category `fix_planned` if the product in question is also listed in a remediation of the category `vendor_fix` with a `date` in the future or no `date` at all. Consequently, the product MUST be removed from the remediation of the category `vendor_fix`. If it was the last product in that remediation, the remediation MUST be removed.
  - The CSAF 2.0 to CSAF 2.1 Converter MUST remove any product from a remediation with the category `none_available` if the product in question is also listed in a remediation of the category `vendor_fix` with a `date` in the past or to the exact same time. If it was the last product in that remediation, the remediation MUST be removed.
  - In any other case, the CSAF 2.0 to CSAF 2.1 Converter MUST preserve the product in the remediation of the category `none_available`.
  - The CSAF 2.0 to CSAF 2.1 Converter MUST output a warning if a remediation was added, deleted or the value of the category was changed, including the products it was changed for.
- The CSAF 2.0 to CSAF 2.1 Converter SHALL provide the JSON path where the warning occurred together with the warning.

A tool MAY implement options to convert other Markdown formats to GitHub-flavored Markdown.

A tool MAY implement an additional, non-default option to output an invalid document that can be fixed afterwards. Solely in this case, any of the rules above MAY be ignored to avoid data loss.

9.1.19 Conformance Clause 19: CSAF Library

A library satisfies the "CSAF Library" conformance profile if the library:

- implements all elements as data structures conforming to the syntax and semantics defined in section 2.2, 2.3, 3, 4 and 5.
- checks all elements according to the patterns provided in the JSON schema.
- has a function that checks version ranges.
- has a function that helps to create version ranges.
- provides for each element functions that allow to create, add, modify and delete that element.
- has a function that reads a CSAF Document into the data structure from a
  - file system.
  - URL.
  - data stream.
- provides function for sorting the keys and sorts the keys automatically on output.
- has a function that outputs the data structure as CSAF Document
  - on the file system.
  - as string.
  - into a data stream.
- has a function to determine the filename according to 5.1 and sets the filename per default when saving a CSAF Document.
- generates a new `product_id` for each new element of type `full_product_name_t` unless an ID is given during the creation.
- generates a new `group_id` for each new element of type `product_group_id_t` unless an ID is given during the creation.
- provides a function to retrieve all elements of type `product_id_t` with its corresponding `full_product_name_t/na` and `full_product_name_t/product_identification_helper`.
- provides a function to retrieve all `product_identification_helper` and their mapping to elements of type `product_id_t`.
- provides a function to retrieve a VEX status mapping for all data, which includes the combination of vulnerability, product, product status and, where necessary according to the profile, the impact statement respectively the action statement.
- provides a function to generate a `full_product_name_t/name` within `branches` through concatenating the `name` values separated by white space of the elements along the path towards this leaf.
- provides a function to generate a combined subpath name for an element of type `subpath_t` through concatenating separated by white space:
  - the `category` with `_` replaced by white space and
  - the `name` value of the product identified by the Product ID in `next_product_reference`.
- provides a function to generate a `full_product_name_t/name` within `product_paths` through concatenating separated by white space:
  - the `name` value of the product identified by the Product ID in `beginning_product_reference` and
  - the combined subpath name for all elements in `subpaths` in their order.
- calculates the CVSS scores and severities for existing data for all CVSS versions.
- validates the CVSS scores and severities for existing data for all CVSS versions.

The library MAY implement an option to retrieve the keys unsorted.

9.1.20 Conformance Clause 20: CSAF Library with Basic Validation

A CSAF Library satisfies the “CSAF Library with Basic Validation” conformance profile if the CSAF Library:

- satisfies the “CSAF Library” conformance profile.
- satisfies the “CSAF Basic Validator” conformance profile.
- validates the CSAF Document before output according to the “CSAF Basic Validator” and presents the validation result accordingly.
- provide a function to validate the data structure in its current state according to the “CSAF Basic Validator” and presents the validation result accordingly.

A CSAF Library does not satisfies the “CSAF Library with Basic Validation” conformance profile if the CSAF Library uses an external library or program for the “CSAF Basic Validator” part and does not enforce its presence.

9.1.21 Conformance Clause 21: CSAF Library with Extended Validation

A CSAF Library satisfies the “CSAF Library with Extended Validation” conformance profile if the CSAF Library:

- satisfies the “CSAF Library” conformance profile.
- satisfies the “CSAF Extended Validator” conformance profile.
- validates the CSAF Document before output according to the “CSAF Extended Validator” and presents the validation result accordingly.
- provide a function to validate the data structure in its current state according to the “CSAF Extended Validator” and presents the validation result accordingly.

A CSAF Library does not satisfies the “CSAF Library with Extended Validation” conformance profile if the CSAF Library uses an external library or program for the “CSAF Extended Validator” part and does not enforce its presence.

9.1.22 Conformance Clause 22: CSAF Library with Full Validation

A CSAF Library satisfies the “CSAF Library with Extended Validation” conformance profile if the CSAF Library:

- satisfies the “CSAF Library” conformance profile.
- satisfies the “CSAF Full Validator” conformance profile.
- validates the CSAF Document before output according to the “CSAF Full Validator” and presents the validation result accordingly.
- provide a function to validate the data structure in its current state according to the “CSAF Full Validator” and presents the validation result accordingly.

A CSAF Library does not satisfies the “CSAF Library with Full Validation” conformance profile if the CSAF Library uses an external library or program for the “CSAF Full Validator” part and does not enforce its presence.

9.1.23 Conformance Clause 23: CSAF Downloader

A program satisfies the “CSAF Downloader” conformance profile if the program:

- conforms to the process defined in section 7.3 by executing all parts that are applicable to the given role.
- supports directory-based and ROLIE-based retrieval.
- is able to execute both steps from section 7.3 separately.
- uses a program-specific HTTP User Agent, e.g. consisting of the name and version of the program.
- satisfies those normative requirements in section 7 that are designated as applying to CSAF Downloaders.

A tool MAY implement an option to store CSAF Documents that fail any of the steps in section 7.3.2.

9.1.24 Conformance Clause 24: CSAF Withdrawer

A program satisfies the "CSAF Withdrawer" conformance profile if the program:

- satisfies the "CSAF Post-Processor" conformance profile.
- keeps the original `/document/tracking/id`.
- adds a new item to the revision history stating the revision metadata of the withdrawal.
- adds the reasoning for withdrawal as specified in section 4.7.
- removes the `/product_tree`.
- removes the `/vulnerabilities`.

> A tool MAY implement an option to additionally remove any element that would hinder the production of a valid CSAF.

9.1.25 Conformance Clause 25: CSAF Superseder

A program satisfies the "CSAF Superseder" conformance profile if the program:

- satisfies the "CSAF Post-Processor" conformance profile.
- keeps the original `/document/tracking/id`.
- adds a new item to the revision history stating the revision metadata of the supersession.
- adds the reasoning for supersession as specified in section 4.8.
- adds the reference to the superseding document as specified in section 4.8.
- removes the `/product_tree`.
- removes the `/vulnerabilities`.

> A tool MAY implement an option to additionally remove any element that would hinder the production of a valid CSAF.

9.1.26 Conformance Clause 26: CSAF RVISC ID Updater

A program satisfies the "CSAF RVISC ID Updater" conformance profile if the program fulfills the two following groups of requirements:

The program:

- satisfies the "CSAF Post-Processor" conformance profile.
- applies the corresponding assignment from [RVISC-M] to each item in `/vulnerabilities[]/ids[]` whose `system_name` is not contained in [RVISC-R].
- applies the corresponding assignment from [RVISC-M] to each item in `/vulnerabilities[]/ids[]` whose `system_name` is contained in [RVISC-R] but the `text` does not conform the entry.
- satisfies the normative requirements given below.

The program MUST provide the following options:

- an option to insert an automatically generated revision history entry detailing the changes applied and make necessary updates to elements in `/document/tracking` (commit mode).
- an option to set selected or all parameters for the commit mode manually which take precedence over the automated generated values.
- an option to do a dry-run which does not apply the changes but just displays them.
- an option to interactively accept or discard changes.
- an option to ignore certain values for `system_name`.
- an option to output all items in `/vulnerabilities[]/ids[]` whose `system_name` is not contained in [RVISC-R].

- an option to output all items in `/vulnerabilities[]/ids[]` whose `system_name` is contained in [RVISC-R] but the `text` does not conform the entry.
- an option to map an existing `system_name` to a new value or apply a transformation to a `text` based on the `system_name` value and a `precondition`.

9.1.27 Conformance Clause 27: CSAF Additional Test

A test satisfies the "CSAF Additional Test" conformance profile if it:

- covers a requirement of the standard.
- operates on a CSAF Document.
- outputs a result object that aligns with result objects provided for tests specified in section 6.
- has a name starting with `AdditionalTest_` followed by a name of the specifying entity conforming to the rules for prefixes of test presets (cf. section 6.4) and an unique name for the test within that entity.
- has its definition specified in the same structure as in section 6.

9.1.28 Conformance Clause 28: CSAF Extension

A JSON object satisfies the "CSAF Extension" conformance profile if it:

- conforms to the syntax and semantics defined in section 2.4.
- validates against the CSAF Extension Content Schema and the schema given through its `$schema` property.
- uses the value of `$id` of its CSAF Extension Schema as the value of its `$schema` property.
- conveys additional information that cannot be conveyed with the CSAF Core elements.
- does not convey any content that can be conveyed with the CSAF Core elements.
- does not contradict the content or purpose of this specification.

9.1.29 Conformance Clause 29: CSAF Extension Schema

A JSON schema satisfies the "CSAF Extension Schema" conformance profile if fulfills the following two groups of requirements:

Firstly, it:

- describes a JSON object satisfying the "CSAF Extension" conformance profile.
- validates against the CSAF Extension Metaschema.
- uses the CSAF Extension Metaschema as the value of its `$schema` property.
- is versioned according to [SemVer].
- has an `$id` that conforms to the `pattern` given for the property `$schema` in section 2.4.4.1.
- does not allow unevaluated properties. It MAY include `patternProperties` if they are typed.

  > `patternProperties` are typed if they can't have more than one `type` and that `type` is given through the schema.

Secondly, it:

- SHOULD avoid unpredictable JSON key names.

  > Implementations usually ignore any additional properties as they are hard to access, process and may imply security risks.

9.1.30 Conformance Clause 30: CSAF Extension Overlay Test

A test satisfies the "CSAF Extension Overlay Test" conformance profile if it:

- is executed only if the specifying CSAF Extension is present.

  > To reuse a test from a different CSAF Extension, it is sufficient to assign it a name according to the rules below and point to the specification of the test it reuses. The referenced specification in such case must be accessible to anyone that can retrieve the referencing CSAF Extension.

- extends or replaces a test specified in this standard in section 6 or a CSAF Additional Test.
- does not adversely affect the purpose of the test in question.
- has a name starting with `Extension_Overlay_Test_` followed by a name of the specifying entity conforming to the rules for prefixes of test presets (cf. section 6.4) and the name of the tests that it replaces.
- has its definition specified in the same structure as in section 6.

The CSAF Extension Overlay Test SHOULD only differ in the paths or the document categories it is applied to.

> This ensures maximum compatibility with other extensions as multiple CSAF Extension can define CSAF Extension Overlay Tests for the same test which are then executed as a union set.

9.1.31 Conformance Clause 31: CSAF Extension Additional Test

A test satisfies the "CSAF Extension Additional Test" conformance profile if it:

- is executed only if the specifying CSAF Extension is present.

  > To reuse a test from a different CSAF Extension, it is sufficient to assign it a name according to the rules below and point to the specification of the test it reuses. The referenced specification in such case must be accessible to anyone that can retrieve the referencing CSAF Extension.

- provides additional checks in the context of the CSAF Extension or the CSAF Document the extension is embedded in.
- has a name starting with `Extension_Additional_Test_` followed by a name of the specifying entity conforming to the rules for prefixes of test presets (cf. section 6.4) and an unique number for the test within that entity.
- has its definition specified in the same structure as in section 6.

9.1.32 Conformance Clause 32: CSAF Extension Test

A test satisfies the "CSAF Extension Test" conformance profile if it:

- satisfies the "CSAF Extension Overlay Test" conformance profile or the "CSAF Extension Additional Test" conformance profile.

9.1.33 Conformance Clause 33: CSAF Extension Specification

A specification satisfies the “CSAF Extension Specification” conformance profile if:

- it defines exactly one extension satisfying the “CSAF Extension” conformance profile.
- it defines exactly one JSON schema satisfying the “CSAF Extension Schema” conformance profile.
- all tests defined in it satisfy the “CSAF Extension Test” conformance profile.
- it contains all tests needed for the validation of the extension.

  > This includes tests that check for prerequisites in the CSAF document, if applicable. Referencing an existing test is considered to fulfill the “contains” requirement, if the referenced test is accessible to anyone that can access the referencing specification.

- all requirements above are fulfilled for the same extension.

9.1.34 Conformance Clause 34: CSAF Extension Bundle

A compilation of artifacts satisfies the “CSAF Extension Bundle” conformance profile if:

- it contains exactly one extension satisfying the “CSAF Extension” conformance profile.
- it contains exactly one JSON schema satisfying the “CSAF Extension Schema” conformance profile.
- it contains exactly one JSON object satisfying the “CSAF Extension Metadata Schema”.
- all necessary test files to implement the “CSAF Extension Test” for this extension.
- has a name that it can be referred to.

9.1.35 Conformance Clause 35: CSAF Extension Package

A compilation of artifacts satisfies the “CSAF Extension Package” conformance profile if:

- it contains exactly one specification satisfying the “CSAF Extension Specification” conformance profile.
- it contains exactly one compilation of artifacts satisfying the “CSAF Extension Bundle” conformance profile.
- has a name that it can be referred to.

9.1.36 Conformance Clause 36: CSAF Extension Collection

A set of artifacts satisfies the “CSAF Extension Collection” conformance profile if it:

- contains one or more named compilation of artifacts that satisfy the “CSAF Extension Package” conformance profile.
- does not contain any compilation of artifacts that looks like a CSAF Extension Package but does not satisfy the “CSAF Extension Package” conformance profile.

Appendix A. Acknowledgments

The following individuals were members of the OASIS CSAF Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

CSAF TC Members:

First Name	Last Name	Company
Aashiq	Ramachandran	Cyware Labs
Alexandre	Dulaunoy	CIRCL
Anthony	Berglas	Cryptsoft Pty Ltd.
Avkash	Kathiriya	Cyware Labs
Bernd	Grobauer	Siemens AG
Chok	Poh	Oracle
Christian	Banse	Fraunhofer-Gesellschaft e.V. - Fraunhofer AISEC
Christoph	Plutte	Ericsson AB
Dan	West	Microsoft
Dave	Tamasi	Google LLC
David	Waltermire	NIST
Denny	Page	Individual
Diane	Morris	Cisco Systems
Dina	Truxius	Federal Office for Information Security (BSI) Germany
Feng	Cao	Oracle
Jane	Ginn	Individual
Jared	Bird	Hewlett Packard Enterprise (HPE)
Jason	Keirstead	Pobal Cyber Ltd
Jean-Didier	Stéfaniak	Dell Technologies
Jeff	Schutt	Cisco Systems
Jennifer	Victor	Dell Technologies
Jessica	Fitzgerald-McKay	National Security Agency
Johannes	Lorenz	Siemens AG
Justin	Corlett	Cryptsoft Pty Ltd.
Justin	Murphy	US DHS Cybersecurity and Infrastructure Security Agency (CISA)
Liming	Wu	Huawei Technologies Co., Ltd.
Martin	Prpic	Red Hat
Michael	Cantonwine	Hewlett Packard Enterprise (HPE)
Michael	Reeder	Dell Technologies
Michael	Schueler	Cisco Systems
Nick	Leali	Cisco Systems
Nicole	Parrish	Mitre Corporation
Oliver	Chang	Google LLC
Omar	Santos	Cisco Systems
Patrick	Maroney	AT&T
Qin	Long	Alibaba Cloud Computing Ltd.
Richard	Struse	Mitre Corporation
Rushyendra	Velamuri	Dell Technologies
Samuel	Cameron	Cisco Systems
Sergii	Demianchuk	Cisco Systems
Shrey	Bagga	Cisco Systems
Sonny	van Lingen	Huawei Technologies Co., Ltd.
Stefan	Hagen	Individual
Tania	Ward	Dell Technologies

First Name	Last Name	Company
Ted	Bedwell	Cisco Systems
Thomas	Proell	Siemens AG
Thomas	Schaffer	Cisco Systems
Thomas	Schmidt	Federal Office for Information Security (BSI) Germany
Tim	Hudson	Cryptsoft Pty Ltd.
Tobias	Limmer	Siemens AG
Tyler	Townes	BlackBerry Limited
Umair	Bukhari	Ericsson AB
Vic	Chung	SAP SE
Vijay	Sarvepalli	Carnegie Mellon University
Vincent	Danen	Red Hat
Vivek	Nair	Microsoft
William	Ho Thiam Hock	Huawei Technologies Co., Ltd.
Xiaoyu	Ge	Huawei Technologies Co., Ltd.
Yogesh	Yadav	Cisco Systems
Zebin	Zhang	Huawei Technologies Co., Ltd.

The following individuals were members of the OASIS CSAF Technical Committee during the creation of the previous version (CSAF 2.0) of this specification and their contributions are gratefully acknowledged:

CSAF TC Members:

First Name	Last Name	Company
Alexandre	Dulaunoy	CIRCL
Anthony	Berglas	Cryptsoft Pty Ltd.
Art	Manion	Carnegie Mellon University
Aukjan	van Belkum	EclecticIQ
Ben	Sooter	Electric Power Research Institute (EPRI)
Bernd	Grobauer	Siemens AG
Bruce	Rich	Cryptsoft Pty Ltd.
Chok	Poh	Oracle
Dan	West	Microsoft
David	Waltermire	NIST
Denny	Page	TIBCO Software Inc.
Duncan	Sparrell	sFractal Consulting LLC
Eric	Johnson	TIBCO Software Inc.
Ethan	Rahn	Arista Networks
Feng	Cao	Oracle
Greg	Scott	Cryptsoft Pty Ltd.
Harold	Booth	NIST
Jason	Masters	TELUS
Jennifer	Victor	Dell
Jessica	Fitzgerald-McKay	National Security Agency
Jonathan	Bitle	Kaiser Permanente
Justin	Corlett	Cryptsoft Pty Ltd.
Kazuo	Noguchi	Hitachi, Ltd.
Kent	Landfield	McAfee
Langley	Rock	Red Hat
Martin	Prpic	Red Hat
Masato	Terada	Hitachi, Ltd.

First Name	Last Name	Company
Mike	Gorski	Cisco Systems
Nicole	Parrish	Mitre Corporation
Omar	Santos	Cisco Systems
Patrick	Maroney	AT&T
Rhonda	Levy	Cisco Systems
Richard	Struse	Mitre Corporation
Ritwik	Ghoshal	Oracle
Robert	Coderre	Accenture
Robert	Keith	Accenture
Stefan	Hagen	Individual
Tania	Ward	Dell
Ted	Bedwell	Cisco Systems
Thomas	Proell	Siemens AG
Thomas	Schmidt	Federal Office for Information Security (BSI) Germany
Tim	Hudson	Cryptsoft Pty Ltd.
Tobias	Limmer	Siemens AG
Tony	Cox	Cryptsoft Pty Ltd.
Vincent	Danen	Red Hat
Will	Rideout	Arista Networks
Xiaoyu	Ge	Huawei Technologies Co., Ltd.

Appendix B. Revision History

Revision	Date	Editor	Changes Made
csaf-v2.1-wd20240124-dev	2024-01-24	Stefan Hagen and Thomas Schmidt	Preparing initial Editor Revision
csaf-v2.1-wd20240228-dev	2024-02-28	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20240327-dev	2024-03-27	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20240424-dev	2024-04-24	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20240529-dev	2024-05-29	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20240626-dev	2024-06-26	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20240731-dev	2024-07-31	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20240828-dev	2024-08-28	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20241030-dev	2024-10-30	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20241127-dev	2024-11-27	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20250129-dev	2025-01-29	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20250226-dev	2025-02-26	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20250326-dev	2025-03-26	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20250430-dev	2025-04-30	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20250528-dev	2025-05-28	Stefan Hagen and Thomas Schmidt	Next Editor Revision
csaf-v2.1-wd20250827-dev	2025-08-27	Stefan Hagen and Thomas Schmidt	Editor Revision containing CSD01PR feedback
csaf-v2.1-wd20250910-dev	2025-09-10	Stefan Hagen and Thomas Schmidt	Editor Revision for CSD02
csaf-v2.1-wd20251029-dev	2025-10-29	Stefan Hagen and Thomas Schmidt	Editor Revision for CSD02
csaf-v2.1-wd20251126-dev	2025-11-26	Stefan Hagen and Thomas Schmidt	Editor Revision for CSD02
csaf-v2.1-wd20251217-dev	2025-12-17	Stefan Hagen and Thomas Schmidt	Editor Revision for CSD02
csaf-v2.1-wd20260128-dev	2026-01-28	Stefan Hagen and Thomas Schmidt	Editor Revision for CSD02
csaf-v2.1-wd20260225-dev	2026-02-25	Stefan Hagen and Thomas Schmidt	Editor Revision for CSD02

Appendix C. Guidance on the Size of CSAF Documents

This appendix provides informative guidance on the size of CSAF documents.

The TC carefully considered all known aspects to provide size limits for CSAF documents for this version of the specification with the result that hard limits SHOULD NOT be enforced. However, since there is the need for guidance to ensure interoperability in the ecosystem, the TC provides a set of soft limits. A CSAF document which exceeds those, can still be valid but it might not be processable for some parties.

All *CSAF consumers* SHOULD be able to process CSAF documents which comply with the limits below. All *CSAF producers* SHOULD NOT produce CSAF documents which exceed those limits.

> If you come across a case where these limits are exceeded, please provide feedback to the TC.

Boolean values are usually represented as a native data type and therefore omitted here. Structures in the `content` object of items of type `/$defs/extensions_t` are not listed in the subsections below as they depend upon on the extension.

C.1 File Size

A CSAF document including any extensions used in the document in the specified JSON format encoded in UTF-8 SHOULD conform to known size limits of current technologies parsing JSON content, e.g.: 150 MiB.

> The CSAF documents observed in the wild expose strongly varying sizes as per the use cases they serve. At least one database technology in wide use for storing CSAF documents rejects insert attempts when the transformed BSON size exceeds 16 megabytes. The BSON format optimizes for accessibility and not size. So, small integers and small strings may incur more overhead in the BSON format than in JSON. In addition, the BSON format adds length information for the entries inside the document, which adds to the size when storing CSAF document content in a BSON format.

C.2 Array Length

An array SHOULD NOT have more than:

- 10 000 items for
  - `/document/acknowledgments`
  - `/document/acknowledgments[]/names`
  - `/document/acknowledgments[]/urls`
  - `/document/tracking/aliases`
  - `/document/x_extensions`
  - `/product_tree/branches[](/branches[])*/product/product_identification_helper/ha`
  - `/product_tree/branches[](/branches[])*/product/product_identification_helper/ha`
  - `/product_tree/branches[](/branches[])*/product/product_identification_helper/pu`
  - `/product_tree/branches[](/branches[])*/product/product_identification_helper/sb`
  - `/product_tree/branches[](/branches[])*/product/product_identification_helper/x_`
  - `/product_tree/branches[](/branches[])*/product/x_extensions`
  - `/product_tree/branches[]/product/product_identification_helper/hashes`
  - `/product_tree/branches[]/product/product_identification_helper/hashes[]/file_ha`
  - `/product_tree/branches[]/product/product_identification_helper/purls`
  - `/product_tree/branches[]/product/product_identification_helper/sbom_urls`
  - `/product_tree/branches[]/product/product_identification_helper/x_generic_uris`
  - `/product_tree/branches[]/product/x_extensions`
  - `/product_tree/full_product_names[]/product_identification_helper/hashes`
  - `/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_`
  - `/product_tree/full_product_names[]/product_identification_helper/purls`

  - /product_tree/full_product_names[]/product_identification_helper/sbom_urls
  - /product_tree/full_product_names[]/product_identification_helper/x_generic_uris
  - /product_tree/full_product_names[]/x_extensions
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/h
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/h
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/p
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/s
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/x
  - /product_tree/product_paths[]/full_product_name/x_extensions
  - /vulnerabilities[]/acknowledgments
  - /vulnerabilities[]/acknowledgments[]/names
  - /vulnerabilities[]/acknowledgments[]/urls
  - /vulnerabilities[]/cwes
  - /vulnerabilities[]/ids
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/decision_point_resources
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/values
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/target_ids
  - /vulnerabilities[]/metrics[]/content/x_extensions
  - /vulnerabilities[]/remediations[]/entitlements
  - /vulnerabilities[]/x_extensions
  - /x_extensions

- 40 000 items for

  - /document/notes
  - /document/references
  - /vulnerabilities[]/involvements
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/references
  - /vulnerabilities[]/notes
  - /vulnerabilities[]/references

- 100 000 for

  - /document/tracking/revision_history
  - /product_tree/branches
  - /product_tree(/branches[])*/branches
  - /product_tree/branches[](/branches[])*/product/product_identification_helper/mo
  - /product_tree/branches[](/branches[])*/product/product_identification_helper/se
  - /product_tree/branches[](/branches[])*/product/product_identification_helper/sk
  - /product_tree/branches[]/product/product_identification_helper/model_numbers
  - /product_tree/branches[]/product/product_identification_helper/serial_numbers
  - /product_tree/branches[]/product/product_identification_helper/skus
  - /product_tree/full_product_names
  - /product_tree/full_product_names[]/product_identification_helper/model_numbers
  - /product_tree/full_product_names[]/product_identification_helper/serial_numbers
  - /product_tree/full_product_names[]/product_identification_helper/skus
  - /product_tree/product_groups[]/product_ids
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/m
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/s
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/s
  - /product_tree/product_paths[]/subpaths
  - /vulnerabilities

- 10 000 000 for

  - /product_tree/product_groups
  - /product_tree/product_paths
  - /vulnerabilities[]/remediations[]/group_ids

- 100 000 000 for

  - /document/notes[]/group_ids
  - /document/notes[]/product_ids
  - /vulnerabilities[]/first_known_exploitation_dates
  - /vulnerabilities[]/first_known_exploitation_dates[]/group_ids
  - /vulnerabilities[]/first_known_exploitation_dates[]/product_ids
  - /vulnerabilities[]/flags
  - /vulnerabilities[]/flags[]/group_ids
  - /vulnerabilities[]/flags[]/product_ids
  - /vulnerabilities[]/involvements[]/group_ids
  - /vulnerabilities[]/involvements[]/product_ids
  - /vulnerabilities[]/metrics
  - /vulnerabilities[]/metrics[]/products
  - /vulnerabilities[]/notes[]/group_ids
  - /vulnerabilities[]/notes[]/product_ids
  - /vulnerabilities[]/product_status/first_affected
  - /vulnerabilities[]/product_status/first_fixed
  - /vulnerabilities[]/product_status/fixed
  - /vulnerabilities[]/product_status/known_affected
  - /vulnerabilities[]/product_status/known_not_affected
  - /vulnerabilities[]/product_status/last_affected
  - /vulnerabilities[]/product_status/recommended
  - /vulnerabilities[]/product_status/under_investigation
  - /vulnerabilities[]/product_status/unknown
  - /vulnerabilities[]/remediations
  - /vulnerabilities[]/remediations[]/product_ids
  - /vulnerabilities[]/threats
  - /vulnerabilities[]/threats[]/group_ids
  - /vulnerabilities[]/threats[]/product_ids

C.3 String Length

A string SHOULD NOT have a length greater than:

- 1000 for

  - /document/acknowledgments[]/names[]
  - /document/acknowledgments[]/organization
  - /document/aggregate_severity/text
  - /document/category
  - /document/distribution/sharing_group/name
  - /document/lang
  - /document/license_expression
  - /document/notes[]/audience
  - /document/notes[]/group_ids[]
  - /document/notes[]/product_ids[]
  - /document/notes[]/title
  - /document/publisher/name
  - /document/source_lang

- /document/title
- /document/tracking/aliases[]
- /document/tracking/generator/engine/name
- /document/tracking/generator/engine/version
- /document/tracking/id
- /document/tracking/revision_history[]/legacy_version
- /document/tracking/revision_history[]/number
- /document/tracking/version
- /product_tree/branches[](/branches[])*/name
- /product_tree/branches[](/branches[])*/product/name
- /product_tree/branches[](/branches[])*/product/product_id
- /product_tree/branches[](/branches[])*/product/product_identification_helper/ha
- /product_tree/branches[](/branches[])*/product/product_identification_helper/ha
- /product_tree/branches[](/branches[])*/product/product_identification_helper/ha
- /product_tree/branches[](/branches[])*/product/product_identification_helper/mo
- /product_tree/branches[](/branches[])*/product/product_identification_helper/se
- /product_tree/branches[](/branches[])*/product/product_identification_helper/sk
- /product_tree/branches[]/name
- /product_tree/branches[]/product/name
- /product_tree/branches[]/product/product_id
- /product_tree/branches[]/product/product_identification_helper/hashes[]/file_ha
- /product_tree/branches[]/product/product_identification_helper/hashes[]/file_ha
- /product_tree/branches[]/product/product_identification_helper/hashes[]/filenam
- /product_tree/branches[]/product/product_identification_helper/model_numbers[]
- /product_tree/branches[]/product/product_identification_helper/serial_numbers[]
- /product_tree/branches[]/product/product_identification_helper/skus[]
- /product_tree/full_product_names[]/name
- /product_tree/full_product_names[]/product_id
- /product_tree/full_product_names[]/product_identification_helper/hashes[]/file_
- /product_tree/full_product_names[]/product_identification_helper/hashes[]/file_
- /product_tree/full_product_names[]/product_identification_helper/hashes[]/filen
- /product_tree/full_product_names[]/product_identification_helper/model_numbers[
- /product_tree/full_product_names[]/product_identification_helper/serial_numbers
- /product_tree/full_product_names[]/product_identification_helper/skus[]
- /product_tree/product_groups[]/group_id
- /product_tree/product_groups[]/product_ids[]
- /product_tree/product_paths[]/full_product_name/name
- /product_tree/product_paths[]/full_product_name/product_id
- /product_tree/product_paths[]/full_product_name/product_identification_helper/h
- /product_tree/product_paths[]/full_product_name/product_identification_helper/h
- /product_tree/product_paths[]/full_product_name/product_identification_helper/h
- /product_tree/product_paths[]/full_product_name/product_identification_helper/m
- /product_tree/product_paths[]/full_product_name/product_identification_helper/s
- /product_tree/product_paths[]/full_product_name/product_identification_helper/s
- /product_tree/product_paths[]/beginning_product_reference
- /product_tree/product_paths[]/subpaths[]/next_product_reference
- /vulnerabilities[]/acknowledgments[]/names[]
- /vulnerabilities[]/acknowledgments[]/organization
- /vulnerabilities[]/cve
- /vulnerabilities[]/cwes[]/id
- /vulnerabilities[]/cwes[]/name
- /vulnerabilities[]/cwes[]/version
- /vulnerabilities[]/flags[]/group_ids[]

  - /vulnerabilities[]/flags[]/product_ids[]
  - /vulnerabilities[]/first_known_exploitation_dates[]/group_ids[]
  - /vulnerabilities[]/ids[]/system_name
  - /vulnerabilities[]/ids[]/text
  - /vulnerabilities[]/involvements[]/contact
  - /vulnerabilities[]/involvements[]/group_ids[]
  - /vulnerabilities[]/metrics[]/content/cvss_v2/vectorString
  - /vulnerabilities[]/metrics[]/content/cvss_v3/vectorString
  - /vulnerabilities[]/metrics[]/content/cvss_v4/vectorString
  - /vulnerabilities[]/metrics[]/content/epss/percentile
  - /vulnerabilities[]/metrics[]/content/epss/probability
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/key
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/namespace
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/values[]/key
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/values[]/name
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/version
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/target_ids[]
  - /vulnerabilities[]/metrics[]/products[]
  - /vulnerabilities[]/notes[]/audience
  - /vulnerabilities[]/notes[]/group_ids[]
  - /vulnerabilities[]/notes[]/product_ids[]
  - /vulnerabilities[]/notes[]/title
  - /vulnerabilities[]/product_status/first_affected[]
  - /vulnerabilities[]/product_status/first_fixed[]
  - /vulnerabilities[]/product_status/fixed[]
  - /vulnerabilities[]/product_status/known_affected[]
  - /vulnerabilities[]/product_status/known_not_affected[]
  - /vulnerabilities[]/product_status/last_affected[]
  - /vulnerabilities[]/product_status/recommended[]
  - /vulnerabilities[]/product_status/under_investigation[]
  - /vulnerabilities[]/product_status/unknown[]
  - /vulnerabilities[]/remediations[]/group_ids[]
  - /vulnerabilities[]/remediations[]/product_ids[]
  - /vulnerabilities[]/threats[]/group_ids[]
  - /vulnerabilities[]/threats[]/product_ids[]
  - /vulnerabilities[]/title

- 10 000 for

  - /document/acknowledgments[]/summary
  - /document/distribution/text
  - /document/publisher/contact_details
  - /document/publisher/issuing_authority
  - /document/references[]/summary
  - /document/tracking/revision_history[]/summary
  - /product_tree/branches[](/branches[])*/product/product_identification_helper/cp
  - /product_tree/branches[](/branches[])*/product/product_identification_helper/pu
  - /product_tree/branches[]/product/product_identification_helper/cpe
  - /product_tree/branches[]/product/product_identification_helper/purls[]
  - /product_tree/full_product_names[]/product_identification_helper/cpe
  - /product_tree/full_product_names[]/product_identification_helper/purls[]
  - /product_tree/product_groups[]/summary
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/c
  - /product_tree/product_paths[]/full_product_name/product_identification_helper/p

  - /vulnerabilities[]/acknowledgments[]/summary
  - /vulnerabilities[]/involvements[]/summary
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/decision_point_resources[]/summary
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/references[]/summary
  - /vulnerabilities[]/references[]/summary
  - /vulnerabilities[]/remediations[]/entitlements[]

- 30 000 for

  - /document/notes[]/text
  - /vulnerabilities[]/notes[]/text
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/description
  - /vulnerabilities[]/metrics[]/content/ssvc_v2/selections[]/values[]/description

- 250 000 for

  - /vulnerabilities[]/remediations[]/details
  - /vulnerabilities[]/remediations[]/restart_required/details
  - /vulnerabilities[]/threats[]/details

C.4 Date

The maximum length of strings representing a temporal value is given by the format specifier. This applies to:

- /document/tracking/current_release_date
- /document/tracking/generator/date
- /document/tracking/initial_release_date
- /document/tracking/revision_history[]/date
- /vulnerabilities[]/disclosure_date
- /vulnerabilities[]/discovery_date
- /vulnerabilities[]/first_known_exploitation_dates[]/date
- /vulnerabilities[]/first_known_exploitation_dates[]/exploitation_date
- /vulnerabilities[]/flags[]/date
- /vulnerabilities[]/involvements[]/date
- /vulnerabilities[]/metrics[]/content/epss/timestamp
- /vulnerabilities[]/metrics[]/content/ssvc_v2/timestamp
- /vulnerabilities[]/remediations[]/date
- /vulnerabilities[]/threats[]/date

C.5 Enum

A string which is an enum has a fixed maximum length given by its longest value.

> Later versions of CSAF might add, modify or delete possible value which could change the longest value. Therefore, this sizes should not be implemented as fixed limits if forward compatibility is desired.

The value of /$schema is a fixed URL, currently pointing to the JSON schema location. It seems to be safe to assume that the length of this value is not greater than 150. This applies to:

- /$schema (59)

For all other values, it seems to be safe to assume that the length of each value is not greater than 50. This applies to:

- /document/csaf_version (3)
- /document/distribution/tlp/label (12)
- /document/notes[]/category (16)

- /document/publisher/category (11)
- /document/references[]/category (8)
- /document/tracking/status (7)
- /document/x_extensions[]/category (13)
- /product_tree/branches[](/branches[])*/category (15)
- /product_tree/branches[](/branches[])*/product/x_extensions[]/category (13)
- /product_tree/branches[]/category (15)
- /product_tree/branches[]/product/x_extensions[]/category (13)
- /product_tree/full_product_names[]/x_extensions[]/category (13)
- /product_tree/product_paths[]/category (21)
- /product_tree/product_paths[]/full_product_name/x_extensions[]/category (13)
- /vulnerabilities[]/flags[]/label (49)
- /vulnerabilities[]/involvements[]/party (11)
- /vulnerabilities[]/involvements[]/status (17)
- /vulnerabilities[]/metrics[]/content/cvss_v2/accessComplexity (6)
- /vulnerabilities[]/metrics[]/content/cvss_v2/accessVector (16)
- /vulnerabilities[]/metrics[]/content/cvss_v2/authentication (8)
- /vulnerabilities[]/metrics[]/content/cvss_v2/availabilityImpact (8)
- /vulnerabilities[]/metrics[]/content/cvss_v2/availabilityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v2/collateralDamagePotential (11)
- /vulnerabilities[]/metrics[]/content/cvss_v2/confidentialityImpact (8)
- /vulnerabilities[]/metrics[]/content/cvss_v2/confidentialityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v2/exploitability (16)
- /vulnerabilities[]/metrics[]/content/cvss_v2/integrityImpact (8)
- /vulnerabilities[]/metrics[]/content/cvss_v2/integrityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v2/remediationLevel (13)
- /vulnerabilities[]/metrics[]/content/cvss_v2/reportConfidence (14)
- /vulnerabilities[]/metrics[]/content/cvss_v2/targetDistribution (11)
- /vulnerabilities[]/metrics[]/content/cvss_v2/version (3)
- /vulnerabilities[]/metrics[]/content/cvss_v3/attackComplexity (4)
- /vulnerabilities[]/metrics[]/content/cvss_v3/attackVector (16)
- /vulnerabilities[]/metrics[]/content/cvss_v3/availabilityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v3/availabilityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/baseSeverity (8)
- /vulnerabilities[]/metrics[]/content/cvss_v3/confidentialityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v3/confidentialityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/environmentalSeverity (8)
- /vulnerabilities[]/metrics[]/content/cvss_v3/exploitCodeMaturity (16)
- /vulnerabilities[]/metrics[]/content/cvss_v3/integrityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v3/integrityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedAttackComplexity (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedAttackVector (16)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedAvailabilityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedConfidentialityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedIntegrityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedPrivilegesRequired (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedScope (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/modifiedUserInteraction (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/privilegesRequired (4)
- /vulnerabilities[]/metrics[]/content/cvss_v3/remediationLevel (13)
- /vulnerabilities[]/metrics[]/content/cvss_v3/reportConfidence (11)
- /vulnerabilities[]/metrics[]/content/cvss_v3/scope (9)
- /vulnerabilities[]/metrics[]/content/cvss_v3/temporalSeverity (8)

- /vulnerabilities[]/metrics[]/content/cvss_v3/userInteraction (8)
- /vulnerabilities[]/metrics[]/content/cvss_v3/version (3)
- /vulnerabilities[]/metrics[]/content/cvss_v4/attackComplexity (4)
- /vulnerabilities[]/metrics[]/content/cvss_v4/attackRequirements (7)
- /vulnerabilities[]/metrics[]/content/cvss_v4/attackVector (8)
- /vulnerabilities[]/metrics[]/content/cvss_v4/Automatable (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/availabilityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/baseSeverity (8)
- /vulnerabilities[]/metrics[]/content/cvss_v4/confidentialityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/exploitMaturity (16)
- /vulnerabilities[]/metrics[]/content/cvss_v4/integrityRequirement (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedAttackComplexity (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedAttackRequirements (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedAttackVector (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedPrivilegesRequired (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedSubAvailabilityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedSubConfidentialityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedSubIntegrityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedUserInteraction (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedVulnAvailabilityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedVulnConfidentialityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/modifiedVulnIntegrityImpact (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/privilegesRequired (4)
- /vulnerabilities[]/metrics[]/content/cvss_v4/providerUrgency (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/Recovery (13)
- /vulnerabilities[]/metrics[]/content/cvss_v4/Safety (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/subAvailabilityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v4/subConfidentialityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v4/subIntegrityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v4/userInteraction (7)
- /vulnerabilities[]/metrics[]/content/cvss_v4/valueDensity (12)
- /vulnerabilities[]/metrics[]/content/cvss_v4/version (3)
- /vulnerabilities[]/metrics[]/content/cvss_v4/vulnAvailabilityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v4/vulnConfidentialityImpact (4)
- /vulnerabilities[]/metrics[]/content/cvss_v4/vulnerabilityResponseEffort (11)
- /vulnerabilities[]/metrics[]/content/cvss_v4/vulnIntegrityImpact (4)
- /vulnerabilities[]/metrics[]/content/qualitative_severity_rating (8)
- /vulnerabilities[]/metrics[]/content/ssvc_v2/schemaVersion (5)
- /vulnerabilities[]/metrics[]/content/x_extensions[]/category (13)
- /vulnerabilities[]/notes[]/category (16)
- /vulnerabilities[]/references[]/category (8)
- /vulnerabilities[]/remediations[]/category (14)
- /vulnerabilities[]/remediations[]/restart_required/category (20)
- /vulnerabilities[]/threats[]/category (14)
- /vulnerabilities[]/x_extensions[]/category (13)
- /x_extensions[]/category (13)

C.6 URI Length

A string with format `uri` SHOULD NOT have a length greater than 20000. This applies to:

- /document/acknowledgments[]/urls[]
- /document/aggregate_severity/namespace

- /document/distribution/tlp/url
- /document/publisher/namespace
- /document/references[]/url
- /document/x_extensions[]/$schema
- /product_tree/branches[](/branches[])*/product/product_identification_helper/sbom_u
- /product_tree/branches[](/branches[])*/product/product_identification_helper/x_gene
- /product_tree/branches[](/branches[])*/product/product_identification_helper/x_gene
- /product_tree/branches[](/branches[])*/product/x_extensions[]/$schema
- /product_tree/branches[]/product/product_identification_helper/sbom_urls[]
- /product_tree/branches[]/product/product_identification_helper/x_generic_uris[]/nam
- /product_tree/branches[]/product/product_identification_helper/x_generic_uris[]/uri
- /product_tree/branches[]/product/x_extensions[]/$schema
- /product_tree/full_product_names[]/product_identification_helper/sbom_urls[]
- /product_tree/full_product_names[]/product_identification_helper/x_generic_uris[]/n
- /product_tree/full_product_names[]/product_identification_helper/x_generic_uris[]/u
- /product_tree/full_product_names[]/x_extensions[]/$schema
- /product_tree/product_paths[]/full_product_name/product_identification_helper/sbom_
- /product_tree/product_paths[]/full_product_name/product_identification_helper/x_ger
- /product_tree/product_paths[]/full_product_name/product_identification_helper/x_ger
- /product_tree/product_paths[]/full_product_name/x_extensions[]/$schema
- /vulnerabilities[]/acknowledgments[]/urls[]
- /vulnerabilities[]/metrics[]/content/ssvc_v2/decision_point_resources[]/uri
- /vulnerabilities[]/metrics[]/content/ssvc_v2/references[]/uri
- /vulnerabilities[]/metrics[]/content/x_extensions[]/$schema
- /vulnerabilities[]/metrics[]/source
- /vulnerabilities[]/references[]/url
- /vulnerabilities[]/remediations[]/url
- /vulnerabilities[]/x_extensions[]/$schema
- /x_extensions[]/$schema

C.7 UUID Length

A string with format `uuid` SHOULD NOT have a length greater than 50. This applies to:

- `/document/distribution/sharing_group/id` (36)

Appendix D. Collapsing Product Paths

The following examples are intended to aid in understanding under which circumstances product paths can be collapsed.

Example 1 (which is not collapsed but collapsible):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version",
              "name": "1.0.0",
              "product": {
                "name": "Example Company Product A 1.0.0",
                "product_id": "CSAFPID-908070601"
              }
            },
            // ...
          ],
          "category": "product_name",
          "name": "Product A"
        },
        {
          "branches": [
            {
              "category": "product_version",
              "name": "2023",
              "product": {
                "name": "Example Company Product B 2023",
                "product_id": "CSAFPID-908070603"
              }
            },
            // ...
          ],
          "category": "product_name",
          "name": "Product B"
        },
        {
          "branches": [
            {
              "category": "product_version",
              "name": "EU",
              "product": {
                "name": "Example Company Product C EU",
                "product_id": "CSAFPID-908070605"
              }
            },
            // ...
          ],
          "category": "product_name",
```



```
          "name": "Product C"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ],
  "product_paths": [
    {
      "beginning_product_reference": "CSAFPID-908070601",
      "full_product_name": {
        "name": "Example Company Product A 1.0.0 installed on Example Company Product
        "product_id": "CSAFPID-908070607"
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070603"
        }
      ]
    },
    // ...
    {
      "beginning_product_reference": "CSAFPID-908070607",
      "full_product_name": {
        "name": "Example Company Product A 1.0.0 installed on Example Company Product
        "product_id": "CSAFPID-908070611"
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070605"
        }
      ]
    },
    // ...
  ]
}
```

Product paths in example 1 above can be collapse as shown in example 2 below.

Example 2 (which is collapsed):

```
"product_tree": {
  "branches": [
    {
      "branches": [
        {
          "branches": [
            {
              "category": "product_version",
              "name": "1.0.0",
```



```
            "product": {
              "name": "Example Company Product A 1.0.0",
              "product_id": "CSAFPID-908070601"
            }
          },
          // ...
        ],
        "category": "product_name",
        "name": "Product A"
      },
      {
        "branches": [
          {
            "category": "product_version",
            "name": "2023",
            "product": {
              "name": "Example Company Product B 2023",
              "product_id": "CSAFPID-908070603"
            }
          },
          // ...
        ],
        "category": "product_name",
        "name": "Product B"
      },
      {
        "branches": [
          {
            "category": "product_version",
            "name": "EU",
            "product": {
              "name": "Example Company Product C EU",
              "product_id": "CSAFPID-908070605"
            }
          },
          // ...
        ],
        "category": "product_name",
        "name": "Product C"
      }
    ],
    "category": "vendor",
    "name": "Example Company"
  }
],
"product_paths": [
  {
    "beginning_product_reference": "CSAFPID-908070601",
    "full_product_name": {
      "name": "Example Company Product A 1.0.0 installed on Example Company Product
      "product_id": "CSAFPID-908070611"
    },
    "subpaths": [
```



```
          {
            "category": "installed_on",
            "next_product_reference": "CSAFPID-908070603"
          },
          {
            "category": "installed_on",
            "next_product_reference": "CSAFPID-908070605"
          }
        ]
      },
      // ...
    ]
  }
```

Example 3 (which is not collapsed nor collapsible without information loss):

```
  "product_tree": {
    "branches": [
      {
        "branches": [
          {
            "branches": [
              {
                "category": "product_version",
                "name": "1.0.0",
                "product": {
                  "name": "Example Company Product A 1.0.0",
                  "product_id": "CSAFPID-908070601",
                  "product_identification_helper": {
                    "cpe": "cpe:2.3:a:example_company:product_a:1.0.0:*:*:*:*:*:*:*"
                  }
                }
              },
              {
                "category": "product_version",
                "name": "1.1.0",
                "product": {
                  "name": "Example Company Product A 1.1.0",
                  "product_id": "CSAFPID-908070602",
                  "product_identification_helper": {
                    "cpe": "cpe:2.3:a:example_company:product_a:1.1.0:*:*:*:*:*:*:*"
                  }
                }
              }
            ],
            "category": "product_name",
            "name": "Product A"
          },
          {
            "branches": [
              {
                "category": "product_version",
```



```
            "name": "2023",
            "product": {
              "name": "Example Company Product B 2023",
              "product_id": "CSAFPID-908070603",
              "product_identification_helper": {
                "cpe": "cpe:2.3:o:example_company:product_b:2023:*:*:*:*:*:*:*"
              }
            }
          },
          {
            "category": "product_version",
            "name": "2024",
            "product": {
              "name": "Example Company Product B 2024",
              "product_id": "CSAFPID-908070604",
              "product_identification_helper": {
                "cpe": "cpe:2.3:o:example_company:product_b:2024:*:*:*:*:*:*:*"
              }
            }
          }
        ],
        "category": "product_name",
        "name": "Product B"
      },
      {
        "branches": [
          {
            "category": "product_version",
            "name": "EU",
            "product": {
              "name": "Example Company Product C EU",
              "product_id": "CSAFPID-908070605",
              "product_identification_helper": {
                "cpe": "cpe:2.3:h:example_company:product_c:eu:*:*:*:*:*:*:*"
              }
            }
          },
          {
            "category": "product_version",
            "name": "US",
            "product": {
              "name": "Example Company Product C US",
              "product_id": "CSAFPID-908070606",
              "product_identification_helper": {
                "cpe": "cpe:2.3:h:example_company:product_c:us:*:*:*:*:*:*:*"
              }
            }
          }
        ],
        "category": "product_name",
        "name": "Product C"
      }
    ],
```



```
      "category": "vendor",
      "name": "Example Company"
    }
  ],
  "product_paths": [
    {
      "beginning_product_reference": "CSAFPID-908070601",
      "full_product_name": {
        "name": "Example Company Product A 1.0.0 installed on Example Company Product
        "product_id": "CSAFPID-908070607",
        "product_identification_helper": {
          "cpe": "cpe:2.3:a:example_company:product_a:1.0.0:*:*:*:*:product_b_2023:*:
        }
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070603"
        }
      ]
    },
    // ...
    {
      "beginning_product_reference": "CSAFPID-908070607",
      "full_product_name": {
        "name": "Example Company Product A 1.0.0 installed on Example Company Product
        "product_id": "CSAFPID-908070611",
        "product_identification_helper": {
          "cpe": "cpe:2.3:a:example_company:product_a:1.0.0:*:*:*:*:product_b_2023:pr
        }
      },
      "subpaths": [
        {
          "category": "installed_on",
          "next_product_reference": "CSAFPID-908070605"
        }
      ]
    },
    // ...
  ]
}
```

Product paths in example 3 cannot be collapsed without information loss. For example, the `cpe` of the product identified by `CSAFPID-908070607` would be lost.

Example 4 (which is not collapsed nor collapsible as products are referenced):

```
{
  // ...
  "product_tree": {
    "branches": [
      {
```



```
      "branches": [
        {
          "branches": [
            {
              "category": "product_version",
              "name": "1.0.0",
              "product": {
                "name": "Example Company Product A 1.0.0",
                "product_id": "CSAFPID-908070601"
              }
            },
            // ...
          ],
          "category": "product_name",
          "name": "Product A"
        },
        {
          "branches": [
            {
              "category": "product_version",
              "name": "2023",
              "product": {
                "name": "Example Company Product B 2023",
                "product_id": "CSAFPID-908070603"
              }
            },
            // ...
          ],
          "category": "product_name",
          "name": "Product B"
        },
        {
          "branches": [
            {
              "category": "product_version",
              "name": "EU",
              "product": {
                "name": "Example Company Product C EU",
                "product_id": "CSAFPID-908070605"
              }
            },
            // ...
          ],
          "category": "product_name",
          "name": "Product C"
        }
      ],
      "category": "vendor",
      "name": "Example Company"
    }
  ],
  // ...
  "product_paths": [
```



```
        {
          "beginning_product_reference": "CSAFPID-908070601",
          "full_product_name": {
            "name": "Example Company Product A 1.0.0 installed on Example Company Produ
            "product_id": "CSAFPID-908070607"
          },
          "subpaths": [
            {
              "category": "installed_on",
              "next_product_reference": "CSAFPID-908070603"
            }
          ]
        },
        // ...
        {
          "beginning_product_reference": "CSAFPID-908070607",
          "full_product_name": {
            "name": "Example Company Product A 1.0.0 installed on Example Company Produ
            "product_id": "CSAFPID-908070611"
          },
          "subpaths": [
            {
              "category": "installed_on",
              "next_product_reference": "CSAFPID-908070605"
            }
          ]
        },
        // ...
      ]
    },
    "vulnerabilities": [
      {
        "product_status": {
          "under_investigation": [
            "CSAFPID-908070607",
            // ...
          ]
        }
      },
      {
        "threats": [
          {
            "category": "target_set",
            "details": "These products are known to be specifically targeted by APT-
00.",
            "product_ids": [
              "CSAFPID-908070607",
              // ...
            ]
          }
        ]
      }
    ]
```



```
}
```

Product paths in example 4 cannot be collapsed as the products inside the path are referenced elsewhere in the document. For example, `CSAFPID-908070607` is also referenced in the product status and threats.

