International Standard

ISO/IEC 20153

Information technology — OASIS Common Security Advisory Framework (CSAF) v2.0 Specification

First edition
2025-02

Reference number
ISO/IEC 20153:2025(en)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by OASIS [as OASIS Common Security Advisory Framework (CSAF) TC] and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

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 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 direct producer: analysis tool which acts as a CSAF producer

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

CSAF extended validator: A CSAF basic validator that additionally performs optional tests.

CSAF full validator: A CSAF extended validator that additionally performs informative tests.

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 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 producer: program that emits output in the CSAF format

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.

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 0

empty object: object that contains no properties

empty string: string that contains no characters, and so has a length of 0

(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

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 characters

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,...) 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

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])

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

[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.

[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.

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.

[CVE]

Common Vulnerability and Exposures (CVE) – The Standard for Information Security Vulnerability Names, MITRE, 1999, https://cve.mitre.org/about/.

[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/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.

[CWE]

Common Weakness Enumeration (CWE) – A Community-Developed List of Software Weakness Types, MITRE, 2005, http://cwe.mitre.org/about/.

[CYCLONEDX13]

CycloneDX Software Bill-of-Material Specification JSON schema version 1.3, cyclonedx.org, May 2021, https://github.com/CycloneDX/specification/blob/1.3/schema/bom-1.3.schema.json.

[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/.

[ISO8601]

Data elements and interchange formats — Information interchange — Representation of dates and times, International Standard, ISO 8601:2004(E), December 1, 2004, https://www.iso.org/standard/40874.html.

[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, 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.

[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.

[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.

[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.

[RFC7464]

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

[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.

[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/.

[SPDX22]

The Software Package Data Exchange (SPDX®) Specification Version 2.2, Linux Foundation and its Contributors, 2020, https://spdx.github.io/spdx-spec/.

[VERS]

vers: a mostly universal version range specifier, Part of the PURL GitHub Project, https://github.com/package-url/purl-spec/blob/version-range-spec/VERSION-RANGE-SPEC.rst.

[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, XX May 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.
```

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

Examples 4321:

```
    Informative examples also use this paragraph style but preceded by the text "Example(s)".
```

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.

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 2.0. 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 defined as a CSAF document is the result of complex orchestration of many players and distinct and partially difficult to play schemas.

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 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:

- Platform Data:
  - Common Platform Enumeration (CPE) Version 2.3 [CPE23-N]
- Vulnerability Scoring:
  - 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
- Vulnerability Classification
  - Common Weakness Enumeration (CWE) [CWE]

        - CWE List: http://cwe.mitre.org/data/index.html
- Classification for Document Distribution
    - Traffic Light Protocol (TLP)
        - Default Definition: https://www.first.org/tlp/

Even though the JSON schema does not 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 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.

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.

3 Schema Elements

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

The CSAF schema Version 2.0 builds on the JSON Schema draft 2020-12 rules.

```
    "$schema": "https://json-schema.org/draft/2020-12/schema"
```

The schema identifier is:

```
    "$id": "https://docs.oasis-open.org/csaf/csaf/v2.0/csaf_json_schema.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 single mandatory property is `document`. The optional two additional properties are `product_tree` and `vulnerabilities`.

3.1 Definitions

The definitions (`$defs`) introduce the following domain specific types into the CSAF language: Acknowledgments (`acknowledgments_t`), Branches (`branches_t`), Full Product Name (`full_product_name_t`), 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`), and Version (`version_t`).

```
    "$defs": {
        "acknowledgments_t": {
            // ...
        },
        "branches_t": {
            // ...
        },
        "full_product_name_t": {
            // ...
        },
        "lang_t": {
            // ...
        },
        "notes_t": {
            // ...
        },
        "product_group_id_t": {
            // ...
        },
        "product_groups_t": {
            // ...
        },
        "product_id_t": {
            // ...
        },
        "products_t": {
            // ...
        },
        "references_t": {
            // ...
        },
        "version_t": {
            // ...
        }
    },
```

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": {
      // ...
      "items": {
        // ...
      }
    },
```

The value type of Acknowledgment is `object` with at least 1 and at most 4 properties. Every such element acknowledges contributions by describing those that contributed. The properties are: `names`, `organization`, `summary`, and `urls`.

```
        "properties": {
          "names": {
            // ...
          },
          "organization": {
            // ...
          },
          "summary": {
            // ...
          },
          "urls": {
            // ...
          }
        }
```

3.1.1.1 Acknowledgments Type - Names

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

Examples 2:

```
    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 3:

```
    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 4:

```
  "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 example 4 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": {
      //...
      "items": {
        // ...
      }
    },
```

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.

```
        "properties": {
          "branches": {
            // ...
          },
          "category": {
            // ...
          },
          "name": {
            // ...
          },
          "product": {
            // ...
          }
        }
```

> `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
legacy
patch_level
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 `legacy` indicates an entry that has reached its end of life.

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

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 SHOULD NOT be used to convey a single version.

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 5:

```
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 6 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 7 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.

   Examples 8 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.

   Tools MAY support this on best effort basis.

   Examples 9 for `name` when using `product_version_range` with vls:

   ```
   <=2
   ```

```
    <4.2
    <V3.0.29
    >=8.1.5
```

3.1.2.4 Branches Type - Product

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

3.1.3 Full Product Name Type

Full Product Name (`full_product_name_t`) with 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": {
      // ...
      "properties": {
        "name": {
          // ...
        },
        "product_id": {
          // ...
        },
        "product_identification_helper": {
          // ...
        }
      }
    },
```

3.1.3.1 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 10:

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

3.1.3.2 Full Product Name Type - Product ID

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

3.1.3.3 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`, `purl`, `sbom_urls`, `serial_numbers`, `skus`, and `x_generic_uris`, one is mandatory.

```
    "product_identification_helper": {
      // ...
      "properties": {
        "cpe": {
          // ...
        },
        "hashes": {
          // ...
        },
        "model_numbers": {
          // ...
        },
        "purl": {
          // ...
        },
        "sbom_urls": {
          // ...
        },
        "serial_numbers": {
          // ...
        },
        "skus": {
          // ...
        },
        "x_generic_uris": {
          // ...
        }
      }
```

3.1.3.3.1 Full Product Name Type - Product Identification Helper - CPE

Common Platform Enumeration representation (`cpe`) of 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.

3.1.3.3.2 Full Product Name Type - Product Identification Helper - Hashes

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

```
    "hashes": {
      // ...
      "items": {
        // ...
      }
    },
```

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 2 mandatory properties `file_hashes` and `filename`.

```
        "properties": {
          "file_hashes": {
            // ...
          },
          "filename": {
            // ...
          }
        }
```

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

```
    "file_hashes": {
      // ...
      "items": {
        // ...
      }
    },
```

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 2 mandatory properties `algorithm` and `value`.

```
        "properties": {
          "algorithm": {
            // ...
          },
          "value": {
            // ...
          }
        }
```

The algorithm of the cryptographic hash representation (`algorithm`) of value type `string` with one or more characters contains the name of the cryptographic hash algorithm used to calculate the value. The default value for `algorithm` is `sha256`.

Examples 11:

```
    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 1.1.1f from 2020-03-31) outputs the following:
>
> ```
> Supported digests:
> -blake2b512                -blake2s256                -md4
> -md5                       -md5-sha1                  -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`) of value type `string` of 32 or more characters with `pattern` (regular expression):

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

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

Examples 12:

```
37df33cb7464da5c7f077f4d56a32bc84987ec1d85b234537c1c1a4d4fc8d09dc29e2e762cb5203677bf849a2855a0283710f1f5fe1d6ce8d5ac85c645d0fcb
3
    4775203615d9534a8bfca96a93dc8b461a489f69124a130d786b42204f3341cc
    9ea4c8200113d49d26505da0e02e2f49055dc078d1ad7a419b32e291c7afebbb84badfbd46dec42883bea0b2a1fa697c
```

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

Examples 13:

```
    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.

3.1.3.3.3 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 full or abbreviated (partial) 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 relationship.

```
    "model_numbers": {
        //...
      "items": {
```

```
      //...
    }
  },
```

Any given model number of value type `string` with at least 1 character represents a full or abbreviated (partial) model number of the component to identify.

> 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 SHOULD begin with the first character of the model number and stop at any point. Characters which SHOULD NOT be matched MUST be replaced by either `?` (for a single character) or `*` (for zero or more characters). Two `*` MUST NOT follow each other.

Examples 14:

```
6RA8096-4MV62-0AA0
6RA801?-??V62-0AA0
IC25T060ATCS05-0
```

3.1.3.3.4 Full Product Name Type - Product Identification Helper - PURL

The package URL (PURL) representation (`purl`) is a `string` of 7 or more characters with `pattern` (regular expression):

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

> The given pattern does not completely evaluate whether a PURL is valid according to the [PURL] 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.

This package URL (PURL) attribute refers to a method for reliably identifying and locating software packages external to this specification. See [PURL] for details.

3.1.3.3.5 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.

```
"sbom_urls": {
    //...
  "items": {
    //...
  }
},
```

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

Examples 15:

```
https://raw.githubusercontent.com/CycloneDX/bom-examples/master/SBOM/keycloak-10.0.2/bom.json
https://swinslow.net/spdx-examples/example4/main-bin-v2
```

3.1.3.3.6 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 full or abbreviated (partial) 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.

```
"serial_numbers": {
    //...
  "items": {
    //...
  }
```

```
    },
```

Any given serial number of value type `string` with at least 1 character represents a full or abbreviated (partial) serial number of the component to identify.

If a part of a serial number of the component to identify is given, it SHOULD begin with the first character of the serial number and stop at any point. Characters which SHOULD NOT be matched MUST be replaced by either `?` (for a single character) or `*` (for zero or more characters).
Two `*` MUST NOT follow each other.

3.1.3.3.7 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 full or abbreviated (partial) stock keeping units.

A list of stock keeping units SHOULD only be used if the list of relationships 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 is or a description how it can be obtained.

> The use of the list of relationships 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.

```
    "skus": {
        //...
      "items": {
        //...
      }
    },
```

Any given stock keeping unit of value type `string` with at least 1 character represents a full or abbreviated (partial) stock keeping unit (SKU) of the component to identify.

> 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 SHOULD begin with the first character of the stock keeping unit and stop at any point. Characters which SHOULD NOT be matched MUST be replaced by either `?` (for a single character) or `*` (for zero or more characters).
Two `*` MUST NOT follow each other.

3.1.3.3.8 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.

```
    "x_generic_uris": {
      // ...
      "items": {
        // ...
      }
    }
```

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

```
        "properties": {
          "namespace": {
            // ...
          },
          "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 16 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.jboss.logging/jboss-
logging@3.4.1.Final?type=jar"
            }
          ]
```

Example 17 linking a component from an SPDX SBOM:

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

3.1.4 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.

Examples 18:

```
    de
    en
    fr
    frc
    jp
```

3.1.5 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": {
      // ...
      "items": {
        // ...
      }
    },
```

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` and `title`.

```
    "properties": {
      "audience": {
        // ...
      },
      "category": {
        // ...
      },
      "text": {
        // ...
      },
    "title": {
      // ...
    }
  }
```

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

Examples 19:

```
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.

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 20:

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

3.1.6 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": {
  // ...
},
```

Examples 21:

```
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.

3.1.7 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": {
  // ...
  "items": {
    // ...
  }
},
```

3.1.8 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": {
  // ...
},
```

Examples 22:

```
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.

3.1.9 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": {
  // ...
  "items": {
    // ...
  }
},
```

3.1.10 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": {
  // ...
  "items": {
    // ...
  }
},
```

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`.

```
"properties": {
  "category": {
    // ...
  },
      "summary": {
        // ...
      },
      "url": {
        // ...
      }
    }
```

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.

3.1.11 Version Type

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

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

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 23:

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

3.1.11.1 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 public release. 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 calculate by integer comparison.

3.1.11.2 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-]+(?:\\.[0-9a-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 regards 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 public release. 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 covert 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 24:

   ```
   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 25:

    ```
    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 26:

       ```
       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 27:

       ```
       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 28:

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

3.2 Properties

These final three subsections document the three properties of a CSAF document. The single mandatory property `document`, as well as the optional properties `product_tree` and `vulnerabilities` in that order.

3.2.1 Document Property

Document level meta-data (`document`) of value type `object` with the 5 mandatory properties Category (`category`), CSAF Version (`csaf_version`), 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 7 optional properties Acknowledgments (`acknowledgments`), Aggregate Severity (`aggregate_severity`), Distribution (`distribution`), Language (`lang`), Notes (`notes`), References (`references`), and Source Language (`source_lang`).

```
"document": {
  // ...
  "properties": {
    "acknowledgments": {
      // ...
    },
    "aggregate_severity" : {
      // ...
    },
    "category": {
```

```
        // ...
      },
      "csaf_version": {
        // ...
      },
      "distribution": {
        // ...
      },
      "lang": {
        // ...
      },
      "notes": {
        // ...
      },
      "publisher": {
        // ...
      },
      "references": {
        // ...
      },
      "source_lang": {
        // ...
      },
      "title": {
        // ...
      },
      "tracking": {
        // ...
      }
    }
  },
```

3.2.1.1 Document Property - Acknowledgments

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

```
  "acknowledgments": {
    // ...
  },
```

3.2.1.2 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.

```
  "aggregate_severity": {
    // ...
    "properties": {
      "namespace": {
        // ...
      },
      "text": {
        // ...
      }
    }
  },
```

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 29:

```
Critical
Important
Moderate
```

3.2.1.3 Document Property - Category

Document category (`category`) with 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.

```
"category": {
  // ...
}
```

Examples 30:

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

3.2.1.4 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:

```
2.0
```

3.2.1.5 Document Property - Distribution

Rules for sharing document (`distribution`) of value type `object` with at least 1 of the 2 properties Text (`text`) and Traffic Light Protocol (TLP) (`tlp`) describes any constraints on how this document might be shared.

```
"distribution": {
  // ...
  "properties": {
    "text": {
      // ...
    },
    "tlp": {
      // ...
    }
  }
},
```

If both values are present, the TLP information SHOULD be preferred as this aids in automation.

3.2.1.5.1 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 31:

```
Copyright 2021, Example Company, All Rights Reserved.
Distribute freely.
Share only on a need-to-know-basis only.
```

3.2.1.5.2 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.

```
"tlp": {
  // ...
  "properties": {
    "label": {
      // ...
    },
    "url": {
      // ...
    }
  }
}
```

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
GREEN
RED
WHITE
```

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 32:

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

3.2.1.6 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.

3.2.1.7 Document Property - Notes

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

```
"notes": {
  // ...
},
```

3.2.1.8 Document Property - Publisher

Publisher (`publisher`) has value type `object` with the mandatory properties Category (`category`), Name (`name`) and Namespace (`namespace`) and provides information on the publishing entity. The 2 other optional properties are: `contact_details` and `issuing_authority`.

```
"publisher": {
  // ...
  "properties": {
    "category": {
      // ...
    },
    "contact_details": {
      // ...
    },
    "issuing_authority": {
      // ...
    },
    "name": {
      // ...
    }
    "namespace": {
      // ...
    }
  }
```

```
    },
```

3.2.1.8.1 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
    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 `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, forwarders, republishers, 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), and product resellers and distributors, including authoritative vendor partners.

3.2.1.8.2 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 33:

```
    Example Company can be reached at contact_us@example.com, or via our website at https://www.example.com/contact.
```

3.2.1.8.3 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.

3.2.1.8.4 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.

Example 34:

```
     BSI
     Cisco PSIRT
     Siemens ProductCERT
```

3.2.1.8.5 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)

Example 35:

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

3.2.1.9 Document Property - References

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

```
"references": {
  // ...
},
```

3.2.1.10 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 and set 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.

3.2.1.11 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 36:

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

3.2.1.12 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`).

```
"tracking": {
  // ...
  "properties": {
    "aliases": {
      // ...
    },
    "current_release_date": {
      // ...
    },
    "generator": {
      // ...
    },
    "id": {
      // ...
    },
    "initial_release_date": {
      // ...
    },
    "revision_history": {
          // ...
        },
        "status": {
          // ...
        },
        "version": {
          // ...
        }
      }
    },
```

3.2.1.12.1 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.

```
    "aliases": {
      // ...
      "items": {
        // ...
      }
    },
```

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 37:

```
    CVE-2019-12345
```

3.2.1.12.2 Document Property - Tracking - Current Release Date

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

3.2.1.12.3 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.

```
        "generator": {
          // ...
          "properties": {
            "date": {
              // ...
            },
            "engine": {
              // ...
            }
          }
        },
```

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.

```
        "engine": {
          // ...
          "properties": {
            "name": {
              // ...
            },
            "version": {
```

```
            // ...
          }
        }
      },
```

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

Examples 38:

```
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 compatible wth 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 39:

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

3.2.1.12.4 Document Property - Tracking - ID

Unique identifier for the document (`id`) of value type `string` with 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 line break.

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 40:

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

> The combination of `/document/publisher/namespace` and `/document/tracking/id` identifies a CSAF document globally unique.

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

3.2.1.12.5 Document Property - Tracking - Initial Release Date

Initial release date (`initial_release_date`) with value type `string` with format `date-time` holds the date when this document was first published.

3.2.1.12.6 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.

```
      "revision_history": {
        // ...
        "items": {
          // ...
        }
      },
```

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`.

```
        "properties": {
          "date": {
            // ...
          },
          "legacy_version": {
            // ...
          },
          "number": {
            // ...
          },
          "summary": {
            // ...
          }
        }
```

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.

3.2.1.12.7 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.

3.2.1.12.8 Document Property - Tracking - Version

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

3.2.2 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 Relationships (`relationships`).

```
    "product_tree": {
      // ...
      "properties": {
        "branches": {
          // ...
        },
        "full_product_names": {
          // ...
        },
        "product_groups": {
          // ...
        },
        "relationships": {
          // ...
        }
      }
    },
```

3.2.2.1 Product Tree Property - Branches

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

3.2.2.2 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.

3.2.2.3 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_groups": {
      // ...
      "items": {
        // ...
      }
    },
```

The product group items are of value type `object` with the 2 mandatory properties Group ID (`group_id`) and Product IDs (`product_ids`) and the optional Summary (`summary`) property.

```
    "properties": {
      "group_id": {
        // ...
      },
      "product_ids": {
        // ...
      },
      "summary": {
        // ...
      }
    }
```

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 41:

```
    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.

3.2.2.4 Product Tree Property - Relationships

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

```
  "relationships": {
    // ...
    "items": {
      // ...
    }
  }
```

The Relationship item is of value type `object` and has four mandatory properties: Relationship category (`category`), Full Product Name (`full_product_name`), Product Reference (`product_reference`), and Relates to Product Reference (`relates_to_product_reference`). The Relationship item establishes a link between two existing `full_product_name_t` elements, allowing the document producer to define a combination of two products that form a new `full_product_name` entry.

```
  "properties": {
    "category": {
      // ...
    },
    "full_product_name": {
      // ...
    },
    "product_reference": {
      // ...
    },
    "relates_to_product_reference": {
      // ...
    }
  }
```

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

Relationship category (`category`) of value type `string` and `enum` defines the category of relationship for the referenced component. The valid 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.

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

Product Reference (`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 first element of the relationship.

Relates to Product Reference (`relates_to_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.

Example 42:

```
"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"
      }
    ],
    "relationships": [
      {
        "product_reference": "CSAFPID-908070601",
        "category": "installed_on",
        "relates_to_product_reference": "CSAFPID-908070602",
        "full_product_name": {
          "product_id": "CSAFPID-908070603",
          "name": "Cisco AnyConnect Secure Mobility Client 2.3.185 installed on Microsoft Windows"
        }
      }
    ]
  }
```

> 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 example 34, it might be the case that `Cisco AnyConnect Secure Mobility Client 4.9.04053"` is only vulnerable when installed on `Microsoft Windows`.

3.2.3 Vulnerabilities Property

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

```
    "vulnerabilities": {
      // ...
      "items": {
        // ...
      }
    }
```

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) (`cwe`), Discovery Date (`discovery_date`), Flags (`flags`), IDs (`ids`), Involvements (`involvements`), Notes (`notes`), Product Status (`product_status`), References (`references`), Release Date (`release_date`), Remediations (`remediations`), Scores (`scores`), Threats (`threats`), and Title (`title`).

```
    "properties": {
      "acknowledgments": {
        // ...
      },
      "cve": {
        // ...
      },
      "cwe": {
        // ...
      },
      "discovery_date": {
        // ...
      },
      "flags": {
        // ...
      },
      "ids": {
        // ...
      },
      "involvements": {
        // ...
      },
      "notes": {
        // ...
      },
      "product_status": {
        // ...
      },
      "references": {
        // ...
      },
      "release_date": {
        // ...
      },
      "remediations": {
        // ...
      },
      "scores": {
        // ...
      },
      "threats": {
        // ...
      },
      "title": {
        // ...
      }
    }
```

3.2.3.1 Vulnerabilities Property - Acknowledgments

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

```
    "acknowledgments": {
      // ...
    },
```

3.2.3.2 Vulnerabilities Property - CVE

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

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

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

3.2.3.3 Vulnerabilities Property - CWE

CWE (`cwe`) of value type `object` with the 2 mandatory properties Weakness ID (`id`) and Weakness Name (`name`) holds the MITRE standard Common Weakness Enumeration (CWE) for the weakness associated. For more information cf. [CWE].

```
    "cwe": {
      // ...
      "properties": {
        "id": {
          // ...
        },
        "name": {
          // ...
        }
      }
    },
```

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

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

and holds the ID for the weakness associated.

Examples 43:

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

The Weakness name (`name`) has value type `string` with 1 or more characters and holds the full name of the weakness as given in the CWE specification.

Examples 44:

```
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')
```

3.2.3.4 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.

3.2.3.5 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.

```
"flags": {
  // ...
  "items": {
    // ...
  }
},
```

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`).

```
"properties": {
  "date": {
    // ...
  },
  "group_ids": {
    // ...
  },
  "label": {
    // ...
  },
  "product_ids": {
    // ...
  }
}
```

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.

3.2.3.6 Vulnerabilities Property - IDs

List of IDs (`ids`) of value type `array` with one 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).

```
"ids": {
  // ...
  "items": {
    // ...
  }
},
```

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.

```
  "properties": {
    "system_name": {
      // ...
    },
    "text": {
      // ...
    }
  }
```

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

Example 45:

```
Cisco Bug ID
GitHub Issue
```

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

Example 46:

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

> 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 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.

3.2.3.7 Vulnerabilities Property - Involvements

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

```
    "involvements": {
      // ...
      "items": {
        // ...
      }
    },
```

Every Involvement item of value type `object` with the 2 mandatory properties Party (`party`), Status (`status`) and the 2 optional properties Date of involvement (`date`) 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`.

```
        "properties": {
          "date": {
            // ...
          },
          "party": {
            // ...
          },
          "status": {
            // ...
          },
          "summary": {
            // ...
          },
        }
```

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

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 3.2.1.8.1).

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.

3.2.3.8 Vulnerabilities Property - Notes

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

```
    "notes": {
      // ...
    },
```

3.2.3.9 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 eight 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`), and Under investigation (`under_investigation`) are all of value type Products (`products_t`).

```
    "product_status": {
      // ...
      "properties": {
        "first_affected": {
          // ...
        },
        "first_fixed": {
          // ...
        },
        "fixed": {
          // ...
        },
        "known_affected": {
          // ...
        },
        "known_not_affected": {
          // ...
        },
        "last_affected": {
          // ...
        },
        "recommended": {
          // ...
        },
        "under_investigation": {
          // ..
        }
      }
    },
```

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[]/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.

3.2.3.10 Vulnerabilities Property - References

Vulnerability references (`references`) of value type References Type (`references_t`) holds a list of references associated with this vulnerability item.

```
    "references": {
      // ...
    },
```

3.2.3.11 Vulnerabilities Property - Release Date

Release date (`release_date`) with value type `string` of format `date-time` holds the date and time the vulnerability was originally released into the wild.

3.2.3.12 Vulnerabilities Property - Remediations

List of remediations (`remediations`) of value type `array` with 1 or more Remediation items of value type `object` contains a list of remediations.

```
    "remediations": {
      // ...
      "items": {
        // ...
      }
    },
```

Every Remediation item of value type `object` with the 2 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`).

```
      "properties": {
        "category": {
          // ...
        },
        "date": {
          // ...
        },
        "details": {
          // ...
        },
        "entitlements": {
          // ...
        },
        "group_ids": {
          // ...
        },
        "product_ids": {
          // ...
        },
        "restart_required": {
          // ...
        },
        "url": {
          // ...
        }
      }
```

3.2.3.12.1 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:

```
    mitigation
    no_fix_planned
    none_available
    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. This value contradicts with the categories `none_available` and `no_fix_planned` for the same product. Therefore, such a combination can't be used in the list of remediations.

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 values `none_available` and `vendor_fix` are mutually exclusive per product.

> An issuing party might choose to use this category to announce that a fix is currently developed. It is recommended that this also includes 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. The values `no_fix_planned` and `vendor_fix` are mutually exclusive per product.

3.2.3.12.2 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.

3.2.3.12.3 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.

3.2.3.12.4 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.

```
            "entitlements": {
                  // ....
                  "items": {
                    // ...
                  }
                },
```

Every Entitlement of the remediation contains any possible vendor-defined constraints for obtaining fixed software or hardware that fully resolves the vulnerability.

3.2.3.12.5 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.

3.2.3.12.6 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.

3.2.3.12.7 Vulnerabilities Property - Remediations - Restart Required

Restart required by remediation (`restart_required`) of value type `object` with the 1 mandatory property Category (`category`) and the optional property Details (`details`) provides information on category of restart is required by this remediation to become effective.

```
      "restart_required": {
        // ...
        "properties": {
          "category": {
            // ...
          }
          "details": {
            // ...
          }
        }
      },
```

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.

3.2.3.12.8 Vulnerabilities Property - Remediations - URL

URL (`url`) of value type `string` with format `uri` contains the URL where to obtain the remediation.

3.2.3.13 Vulnerabilities Property - Scores

List of scores (`scores`) of value type `array` with 1 or more items of type score holds a list of score objects for the current vulnerability.

```
"scores": {
  // ...
  "items": {
    // ...
  }
},
```

Value type of every such Score item is `object` with the mandatory property `products` and the optional properties `cvss_v2` and `cvss_v3` specifies information about (at least one) score of the vulnerability and for which products the given value applies. Each Score item has at least 2 properties.

```
    "properties": {
      "cvss_v2": {
        // ...
      },
      "cvss_v3": {
        "oneOf": [
          // ...
        ]
      }
      "products": {
        // ...
      }
    }
```

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.

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.

Product IDs (`products`) of value type `products_t` with 1 or more items indicates for which products the given scores apply. A score 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).

3.2.3.14 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.

```
"threats": {
  // ...
  "items": {
    // ...
  }
},
```

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`).

```
"properties": {
  "category": {
    // ...
```

```
      }
      "date": {
        // ...
      },
      "details": {
        // ...
      },
      "group_ids": {
        // ...
      },
      "product_ids": {
        // ...
      }
    }
```

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 "Exploitability" 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.

3.2.3.15 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.

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, upcoming and future profiles defined in the CSAF standard.
5. 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.
6. Local or private profiles MAY exist and tools MAY choose to support them.
7. 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:
  - `/document/category`
  - `/document/csaf_version`
  - `/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 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. This does not differentiate between underscore, dash or whitespace. 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.

- The value of `/document/category` SHALL be `csaf_security_advisory`.

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 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`.

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 lower case.
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 47:

```
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 48:

```
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.

6 Tests

The following three subsections list a number of tests which all will have a short description and an excerpt of an example which fails the test.

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:

```
/product_tree/product_groups[]/product_ids[]
/product_tree/relationships[]/product_reference
/product_tree/relationships[]/relates_to_product_reference
/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[]/remediations[]/product_ids[]
/vulnerabilities[]/scores[]/products[]
/vulnerabilities[]/threats[]/product_ids[]
```

Example 49 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/relationships[]/full_product_name/product_id
```

Example 50 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 relationships (`/product_tree/relationships`) it MUST be tested that the `product_id` does not end up in a circle.

The relevant path for this test is:

```
  /product_tree/relationships[]/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 relationship item is used as `product_reference` or `relates_to_product_reference`. Only for those which fulfill this condition it is necessary to run the full check following the references.

Example 51 which fails the test:

```
  "product_tree": {
    "full_product_names": [
      {
        "product_id": "CSAFPID-9080700",
        "name": "Product A"
      }
    ],
    "relationships": [
      {
        "category": "installed_on",
        "full_product_name": {
          "name": "Product B",
          "product_id": "CSAFPID-9080701"
        },
        "product_reference": "CSAFPID-9080700",
        "relates_to_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 `/$defs/product_group_id_t` which is not inside a Product Group (`/product_tree/product_groups[]`) and therefore reference an element within the `product_tree` it MUST be tested that the Product Group element with the matching `group_id` exists. The same applies for all items of elements of type `/$defs/product_groups_t`.

The relevant paths for this test are:

```
  /vulnerabilities[]/remediations[]/group_ids
  /vulnerabilities[]/threats[]/group_ids
```

Example 52 which fails the test:

```
  "product_tree": {
    "full_product_names": [
      {
        "product_id": "CSAFPID-9080700",
        "name": "Product A"
      }
    ]
  },
"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 53 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 member of contradicting product status groups. The sets formed by the contradicting groups within one vulnerability item MUST be pairwise disjoint.

Contradiction groups are:

- 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[]
```

> 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.

Example 54 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 member of more than one CVSS-Vectors with the same version.

The relevant path for this test is:

```
/vulnerabilities[]/scores[]
```

Example 55 which fails the test:

```
  "product_tree": {
    "full_product_names": [
      {
        "product_id": "CSAFPID-9080700",
        "name": "Product A"
      }
    ]
  },
  "vulnerabilities": [
    {
      "scores": [
        {
          "products": [
            "CSAFPID-9080700"
          ],
          "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"
          ],
          "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"
          }
        }
      ]
    }
  ]
```

> Two CVSS v3.1 scores are given for `CSAFPID-9080700`.

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[]/scores[]/cvss_v2
  /vulnerabilities[]/scores[]/cvss_v3
```

Example 56 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 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[]/scores[]/cvss_v2/baseScore
  /vulnerabilities[]/scores[]/cvss_v2/temporalScore
  /vulnerabilities[]/scores[]/cvss_v2/environmentalScore
  /vulnerabilities[]/scores[]/cvss_v3/baseScore
```

```
/vulnerabilities[]/scores[]/cvss_v3/baseSeverity
/vulnerabilities[]/scores[]/cvss_v3/temporalScore
/vulnerabilities[]/scores[]/cvss_v3/temporalSeverity
/vulnerabilities[]/scores[]/cvss_v3/environmentalScore
/vulnerabilities[]/scores[]/cvss_v3/environmentalSeverity
```

Example 57 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 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[]/scores[]/cvss_v2
/vulnerabilities[]/scores[]/cvss_v3
```

Example 58 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 CVSS vector differs from values of the properties `attackVector`, `scope` and `availabilityImpact`.

> A tool MAY overwrite contradicting values according to the `vectorString` as quick fix.

6.1.11 CWE

It MUST be tested that given CWE exists and is valid.

The relevant path for this test is:

```
/vulnerabilities[]/cwe
```

Example 59 which fails the test:

```
"cwe": {
  "id": "CWE-79",
  "name": "Improper Input Validation"
}
```

> The `CWE-79` exists. However, its name is `Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')`.

6.1.12 Language

For each element of type `/$defs/language_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 60 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 given PURL is valid.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/purl
/product_tree/full_product_names[]/product_identification_helper/purl
/product_tree/relationships[]/full_product_name/product_identification_helper/purl
```

Example 61 which fails the test:

```
"product_tree": {
  "full_product_names": [
    {
      "name": "Product A",
      "product_id": "CSAFPID-9080700",
      "product_identification_helper": {
        "purl": "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`.

The relevant path for this test is:

```
  /document/tracking/revision_history
```

Example 62 which fails the test:

```
"revision_history": [
  {
    "date": "2021-07-22T10:00:00.000Z",
    "number": "2",
    "summary": "Second version."
  },
  {
    "date": "2021-07-23T10:00:00.000Z",
    "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`.

The relevant path for this test is:

```
/document/source_lang
```

Example 63 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.

6.1.16 Latest Document Version

It MUST be tested that document version has the same value as the the `number` in the last item of Revision History when it is sorted ascending by `date`. 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 64 which fails the test:

```
"tracking": {
  // ...
  "revision_history": [
    {
      "date": "2021-07-21T09:00:00.000Z",
      "number": "1",
      "summary": "Initial version."
    },
    {
      "date": "2021-07-21T10:00:00.000Z",
      "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`.

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 65 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`.

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 66 which fails the test:

```
"tracking": {
  // ...
  "revision_history": [
    {
      "date": "2021-05-17T10:00:00.000Z",
      "number": "0",
      "summary": "First draft"
    },
    {
      "date": "2021-07-21T10:00:00.000Z",
      "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`.

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 67 which fails the test:

```
"revision_history": [
  {
    "date": "2021-04-22T10:00:00.000Z",
    "number": "1.0.0-rc",
    "summary": "Release Candidate for initial version."
  },
  {
    "date": "2021-04-23T10:00:00.000Z",
    "number": "1.0.0",
    "summary": "Initial version."
  }
]
```

> The revision history contains an item which has a `number` that indicates that this is pre-release.

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 68 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`.

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`. 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 69 which fails the test:

```
"revision_history": [
  {
    "date": "2021-04-22T10:00:00.000Z",
    "number": "1",
    "summary": "Initial version."
  },
  {
    "date": "2021-07-21T10:00:00.000Z",
    "number": "3",
    "summary": "Some other changes."
  }
]
```

> The item for version `2` is missing.

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 70 which fails the test:

```
"revision_history": [
  {
    "date": "2021-07-20T10:00:00.000Z",
    "number": "1",
    "summary": "Initial version."
  },
  {
    "date": "2021-07-21T10:00:00.000Z",
    "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 71 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 72 which fails the test:

```
"vulnerabilities": [
  {
    "involvements": [
      {
        "date": "2021-04-23T10:00:00.000Z",
        "party": "vendor",
        "status": "completed"
      },
      {
        "date": "2021-04-23T10:00:00.000Z",
        "party": "vendor",
        "status": "in_progress",
        "summary": "The vendor has released a mitigation and is working to fully resolve the issue."
      }
    ]
  }
]
```

> 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 hashes.

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/product/product_identification_helper/hashes[]/file_hashes
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/relationships[]/full_product_name/product_identification_helper/hashes[]/file_hashes
```

Example 73 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": "026a37919b182ef7c63791e82c9645e2f897a3f0b73c7a6028c7febf62e93838"
                },
                {
                  "algorithm": "sha256",
                  "value": "0a853ce2337f0608489ac596a308dc5b7b19d35a52b10bf31261586ac368b175"
                }
              ],
              "filename": "product_a.so"
            }
          ]
        }
      }
    ]
  }
```

> The hash algorithm `sha256` is used two times in one item of 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, whitespace, and underscore characters are removed from the values on both sides before the match. Also the value MUST NOT start with the reserved prefix `csaf_` except if the value is `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 2.0, the test must be skipped for the following values in `/document/category`:
>
> ```
> csaf_base
> csaf_security_incident_response
> csaf_informational_advisory
> csaf_security_advisory
> csaf_vex
> ```

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 74 for currently prohibited values:

```
Csaf_a
Informational Advisory
security-incident-response
Security      Advisory
veX
V_eX
```

Example 75 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 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 76 which fails the test:

```
"notes": [
  {
    "category": "legal_disclaimer",
    "text": "The CSAF document is provided to You \"AS IS\" and \"AS AVAILABLE\" and with all faults and defects
without warranty of any kind.",
    "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 77 which fails the test:

```
"references": [
  {
    "category": "self",
    "summary": "The canonical URL.",
    "url": "https://example.com/security/data/csaf/2021/OASIS_CSAF_TC-CSAF_2_0-2021-6-1-27-02-01.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 value for `/document/category` is:

```
csaf_informational_advisory
```

The relevant path for this test is:

```
/vulnerabilities
```

Example 78 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
```

The relevant path for this test is:

```
/product_tree
```

Example 79 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
```

The relevant path for this test is:

```
/vulnerabilities[]/notes
```

Example 80 which fails the test:

```
"vulnerabilities": [
  {
    "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 value for /document/category is:

```
csaf_security_advisory
```

The relevant path for this test is:

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

Example 81 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 82 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 83 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 84 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 85 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 provided.",
        "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
```

The relevant path for this test is:

```
/vulnerabilities
```

Example 86 which fails the test:

```
{
  "document": {
    // ...
  },
  "product_tree": [
    // ...
  ]
}
```

> The element `/vulnerabilities` does not exist.

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 87 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 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 88 which fails the test:

```
    "remediations": [
      {
        "category": "no_fix_planned",
        "details": "These products are end-of-life. Therefore, no fix will be provided."
      }
    ]
```

> 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 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 89 which fails the test:

```
  "tracking": {
    // ...
    "revision_history": [
      {
        "date": "2021-07-21T09:00:00.000Z",
        "number": "1.0.0",
        "summary": "Initial version."
      },
      {
        "date": "2021-07-21T10:00:00.000Z",
        "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 lower case, the value of `name` does not contain any of the following strings:
>
> ```
> <
> <=
> >
> >=
> after
> all
> before
> earlier
> later
> prior
> versions
> ```

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/name
```

Example 90 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 91 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 3.2.3.5). 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 92 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 for `CSAFPID-9080700` - one indirect through `CSAFGID-0001` and one direct.

6.2 Optional Tests

Optional 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/relationships[]/full_product_name/product_id
```

Example 93 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 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 94 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 Score

For each Product ID (type `/$defs/product_id_t`) in the Product Status groups Affected it MUST be tested that a score 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 95 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 score 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 96 which fails the test:

```
"revision_history": [
  {
    "date": "2021-04-23T10:00:00.000Z",
    "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.

The relevant path for this test is:

```
/document/tracking/initial_release_date
```

Example 97 which fails the test:

```
"tracking": {
  // ...
  "initial_release_date": "2021-04-22T10:00:00.000Z",
  "revision_history": [
    {
      "date": "2021-05-06T10:00:00.000Z",
      "number": "1",
      "summary": "Initial version."
    },
    {
      "date": "2021-07-21T11:00:00.000Z",
      "number": "2",
      "summary": "Second version."
    }
  ],
  // ...
}
```

> The initial release date `2021-04-22T10:00:00.000Z` is older than `2021-05-06T10:00:00.000Z` 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.

The relevant path for this test is:

```
/document/tracking/current_release_date
```

Example 98 which fails the test:

```
"tracking": {
  "current_release_date": "2021-05-06T10:00:00.000Z",
  // ...
  "revision_history": [
    {
        "date": "2021-05-06T10:00:00.000Z",
        "number": "1",
        "summary": "Initial version."
      },
      {
        "date": "2021-07-21T11:00:00.000Z",
        "number": "2",
        "summary": "Second version."
      }
    ],
    // ...
  }
```

> The current release date `2021-05-06T10:00:00.000Z` is older than `2021-05-23T1100:00.000Z` 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 99 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[]/file_hashes
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/relationships[]/full_product_name/product_identification_helper/hashes[]/file_hashes
```

Example 100 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[]/file_hashes
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes
/product_tree/relationships[]/full_product_name/product_identification_helper/hashes[]/file_hashes
```

Example 101 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

It MUST be tested that `/document/distribution/tlp/label` is present and valid.

> TLP labels support the machine-readability and automated distribution.

The relevant path for this test is:

```
/document/distribution/tlp/label
```

Example 102 which fails the test:

```
    "distribution": {
      "text": "Distribute freely."
    }
```

> The CSAF document has no TLP label.

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 103 which fails the test:

```
"document": {
  // ...
  "references": [
    {
      "category": "self",
      "summary": "A non-canonical URL.",
      "url": "https://example.com/security/data/csaf/2021/OASIS_CSAF_TC-CSAF_2.0-2021-6-2-11-01_1.json"
    }
  ],
  // ...
  "tracking": {
    // ...
    "id": "OASIS_CSAF_TC-CSAF_2.0-2021-6-2-11-01",
    // ...
    "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 is present and set.

The relevant path for this test is:

```
/document/lang
```

Example 104 which fails the test:

```
"document": {
  "category": "csaf_base",
  "csaf_version": "2.0",
  "publisher": {
    // ...
  },
  // ...
}
```

> The document language is not defined.

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 105 which fails the test:

```
"document": {
  "csaf_version": "2.0",
  "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/language_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 106 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/language_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 107 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/relationships[]/full_product_name
```

Example 108 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 109 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` conforms the vers specification.

> To implement this test it is deemed sufficient that the value of `name` matches the following regex:
>
> ```
> ^vers:[a-z\\.\\-\\+][a-z0-9\\.\\-\\+]*/.+
> ```

The relevant paths for this test are:

```
/product_tree/branches[](/branches[])*/name
```

Example 110 which fails the test:

```
        "branches": [
          {
            "category": "product_version_range",
            "name": ">4.2",
            // ...
          }
        ]
```

> The version range `>4.2` is a valid vsl 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 environmental 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 `scores` element.

The relevant path for this test is:

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

Example 111 which fails the test:

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
```

```
        "name": "Product A"
      }
    ]
  },
  "vulnerabilities": [
    {
      "product_status": {
        "fixed": [
          "CSAFPID-9080700"
        ]
      },
      "scores": [
        {
          "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 `modifiedIntegrityImpact`, `modifiedAvailabilityImpact`, `modifiedConfidentialityImpact` nor the corresponding attributes in the `vectorString` have been set.

> A tool MAY set the properties `modifiedIntegrityImpact`, `modifiedAvailabilityImpact`, `modifiedConfidentialityImpact` accordingly and compute the `environmentalScore` as quick fix.

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.

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 sets `additionalProperties` to `false` for every key of type `object`.

Example 112 which fails the test:

```
  "document": {
    "category": "csaf_base",
    "csaf_version": "2.0",
    "custom_property": "any",
    // ...
  }
```

> The key `custom_property` is not defined in the JSON schema.

> A tool MAY remove such keys as a quick fix.

6.3 Informative Test

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 scores which contains the `cvss_v2` object it MUST be tested that is not the only scoring item present. The test SHALL pass if a second scoring object is available.

The relevant path for this test is:

```
  /vulnerabilities[]/scores
```

Example 113 which fails the test:

```
"product_tree": {
  "full_product_names": [
    {
      "product_id": "CSAFPID-9080700",
      "name": "Product A"
    }
  ]
},
"vulnerabilities": [
  {
    "scores": [
      {
        "products": [
          "CSAFPID-9080700"
        ],
        "cvss_v2": {
          "version": "2.0",
          "vectorString": "AV:N/AC:L/Au:N/C:C/I:C/A:C",
          "baseScore": 10
        }
      }
    ]
  }
]
```

> There is only a CVSS v2 score given for `CSAFPID-9080700`.

Recommendation:

It is recommended to (also) use the CVSS v3.1.

6.3.2 Use of CVSS v3.0

For each item in the list of scores which contains the `cvss_v3` object it MUST be tested that CVSS v3.0 is not used.

The relevant paths for this test are:

```
/vulnerabilities[]/scores[]/cvss_v3/version
/vulnerabilities[]/scores[]/cvss_v3/vectorString
```

Example 114 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 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 115 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 the CWE is given.

The relevant path for this test is:

```
/vulnerabilities[]/cwe
```

Example 116 which fails the test:

```
"vulnerabilities": [
  {
    "cve": "CVE-2019-0708",
    "title": "BlueKeep"
  }
]
```

> The CWE number is not 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[]/file_hashes[]/value
/product_tree/full_product_names[]/product_identification_helper/hashes[]/file_hashes[]/value
/product_tree/relationships[]/full_product_name/product_identification_helper/hashes[]/file_hashes[]/value
```

Example 117 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[]/namespace
/product_tree/branches[]/product/product_identification_helper/x_generic_uris[]/uri
/product_tree/branches[](/branches[])*/product/product_identification_helper/sbom_urls[]
/product_tree/branches[](/branches[])*/product/product_identification_helper/x_generic_uris[]/namespace
/product_tree/branches[](/branches[])*/product/product_identification_helper/x_generic_uris[]/uri
/product_tree/full_product_names[]/product_identification_helper/sbom_urls[]
/product_tree/full_product_names[]/product_identification_helper/x_generic_uris[]/namespace
/product_tree/full_product_names[]/product_identification_helper/x_generic_uris[]/uri
/product_tree/relationships[]/full_product_name/product_identification_helper/sbom_urls[]
/product_tree/relationships[]/full_product_name/product_identification_helper/x_generic_uris[]/namespace
/product_tree/relationships[]/full_product_name/product_identification_helper/x_generic_uris[]/uri
/vulnerabilities[]/acknowledgments[]/urls[]
/vulnerabilities[]/references[]/url
/vulnerabilities[]/remediations[]/url
```

Example 118 which fails the test:

```
    "references": [
      {
        "summary": "A URL that does not resolve with HTTP status code in the interval between (including) 200 and
(excluding) 400.",
        "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 119 which fails the test:

```
    "references": [
      {
        "category": "self",
        "summary": "A URL that does not resolve with HTTP status code in the interval between (including) 200 and
(excluding) 400.",
        "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 not document language is set. It SHALL fail it 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/relationships[]/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 120 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 121 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-908070` 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 122 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 123 which fails the test:

```
      "branches": [
        {
          "category": "product_version",
          "name": "v4.2",
          // ...
        }
      ]
```

> The product version starts with a `v`.

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:WHITE

If the CSAF document is labeled TLP:WHITE, 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 and TLP:RED

CSAF documents labeled TLP:AMBER 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:WHITE, 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.

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 124 Minimal with ROLIE document:

```
{
  "canonical_url": "https://www.example.com/.well-known/csaf/provider-metadata.json",
  "distributions": [
    {
      "rolie":{
        "feeds": [
          {
            "summary":"All TLP:WHITE advisories of Example Company.",
            "tlp_label": "WHITE",
            "url": "https://www.example.com/.well-known/csaf/feed-tlp-white.json"
          }
        ]
      }
    }
  ],
  "last_updated": "2021-07-12T20:20:56.169Z",
  "list_on_CSAF_aggregators": true,
  "metadata_version": "2.0",
  "mirror_on_CSAF_aggregators": true,
  "public_openpgp_keys": [
    {
      "fingerprint": "8F5F267907B2C4559DB360DB2294BA7D2B2298B1",
      "url": "https://keys.example.net/vks/v1/by-fingerprint/8F5F267907B2C4559DB360DB2294BA7D2B2298B1"
    }
  ],
  "publisher": {
    "category": "vendor",
    "name": "Example Company ProductCERT",
    "namespace":"https://psirt.example.com"
  },
  "role": "csaf_trusted_provider"
}
```

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. At the time of this writing, the `CSAF` field is in the process of being officially added.

Examples 125:

```
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
CSAF: https://www.example.com/.well-known/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 merges or acquisitions. However, this SHOULD NOT be done and removed as soon as possible. If one of the URLs fulfills requirement 9, this MUST be used 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. The use of the scheme "HTTPS" is required. See [RFC8615] for more details.

Example 126:

```
https://www.example.com/.well-known/csaf/provider-metadata.json
```

7.1.10 Requirement 10: DNS path

The DNS record `csaf.data.security.domain.tld` SHALL resolve as a web server which serves directly the `provider-metadata.json` according to requirement 7. 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 127:

```
2021
2020
```

7.1.12 Requirement 12: index.txt

The index.txt file within MUST provide a list of all filenames of CSAF documents which are located in the sub-directories with their filenames.

Example 128:

```
2020/example_company_-_2020-yh4711.json
2019/example_company_-_2019-yh3234.json
2018/example_company_-_2018-yh2312.json
```

> This can be used to download all CSAF documents.

7.1.13 Requirement 13: changes.csv

The file changes.csv 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.

Example 129:

```
"2020/example_company_-_2020-yh4711.json","2020-07-01T10:09:07Z"
"2018/example_company_-_2018-yh2312.json","2020-07-01T10:09:01Z"
"2019/example_company_-_2019-yh3234.json","2019-04-17T15:08:41Z"
"2018/example_company_-_2018-yh2312.json","2019-03-01T06:01:00Z"
```

7.1.14 Requirement 14: Directory listings

Directory listing SHALL be enabled to support manual navigation.

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. At least one of the feeds

- TLP:WHITE
- TLP:GREEN
- unlabeled

MUST exist. Each ROLIE feed document MUST be a JSON file that conforms with [RFC8322].

Example 130:

```
{
  "feed": {
    "id": "example-csaf-feed-tlp-white",
      "title": "Example CSAF feed (TLP:WHITE)",
      "link": [
        {
          "rel": "self",
          "href": "https://psirt.domain.tld/advisories/csaf/feed-tlp-white.json"
        }
      ],
      "category": [
        {
          "scheme": "urn:ietf:params:rolie:category:information-type",
          "term": "csaf"
        }
      ],
      "updated": "2021-01-01T12:00:00.000Z",
      "entry": [
        {
          "id": "2020-ESA-001",
          "title": "Example Security Advisory 001",
          "link": [
            {
              "rel": "self",
              "href": "https://psirt.domain.tld/advisories/csaf/2020/2020-ESA-001.json"
            },
            {
              "rel": "hash",
              "href": "https://psirt.domain.tld/advisories/csaf/2020/2020-ESA-001.json.sha512"
            },
            {
              "rel": "signature",
              "href": "https://psirt.domain.tld/advisories/csaf/2020/2020-ESA-001.json.asc"
            }
          ],
          "published": "2021-01-01T11:00:00.000Z",
          "updated": "2021-01-01T12:00:00.000Z",
          "summary": {
            "content": "Vulnerabilities fixed in ABC 0.0.1"
          },
          "content": {
            "type": "application/json",
            "src": "https://psirt.domain.tld/advisories/csaf/2020/2020-ESA-001.json"
          },
          "format": {
            "schema": "https://docs.oasis-open.org/csaf/csaf/v2.0/csaf_json_schema.json",
            "version": "2.0"
          }
        }
      ]
    }
  }
```

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.

Example 131:

```
  {
    "service": {
      "workspace": [
        {
          "title": "Public CSAF feed",
          "collection": [
            {
              "title": "Example CSAF feed (TLP:WHITE)",
              "href": "https://psirt.domain.tld/advisories/csaf/feed-tlp-white.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]. 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

  Example 132:

  ```
  CPU
  Firewall
  Monitor
  PLC
  Printer
  Router
  Sensor
  Server
  ```

- areas or sectors, the products are used in

  Example 133:

  ```
  Chemical
  Commercial
  Communication
  Critical Manufacturing
  Dams
  Energy
  Healthcare
  Water
  ```

- any other categorization useful to the consumers

Example 134:

```
{
  "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 SHOULD NOT be used.

Example 135:

```
File name of CSAF document: example_company_-_2019-yh3234.json
File name of SHA-256 hash file: example_company_-_2019-yh3234.json.sha256
File name of SHA-512 hash file: example_company_-_2019-yh3234.json.sha512
```

The file content SHALL start with the first byte of the hexadecimal hash value. Any subsequent data (like a filename) which is optional SHALL be separated by at least one space.

Example 136:

```
ea6a209dba30a958a78d82309d6cdcc6929fcb81673b3dc4d6b16fac18b6ff38  example_company_-_2019-yh3234.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. See [RFC4880] for more details.

Example 137:

```
File name of CSAF document: example_company_-_2019-yh3234.json
File name of signature file: example_company_-_2019-yh3234.json.asc
```

If a ROLIE feed exists, each signature file MUST be listed in it as described in requirement 15.

7.1.20 Requirement 20: Public OpenPGP Key

The public part of the OpenPGP key used to sign the CSAF documents MUST be available. 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`.

> Suggested locations to store the `aggregator.json` are:
>
> - https://www.example.com/.well-known/csaf-aggregator/aggregator.json
> - https://domain.tld/security/data/aggregator/csaf/aggregator.json
> - https://psirt.domain.tld/advisories/aggregator/csaf/aggregator.json
> - https://domain.tld/security/aggregator/csaf/aggregator.json