OpenEOX Core Schema Version 1.0

Committee Specification Draft 01/WD

21 January 2026

This stage:

https://docs.oasis-open.org/openeox/eox-core/v2.1/csd01/eox-core-v1.0-csd01.md (Authoritative)
https://docs.oasis-open.org/openeox/eox-core/v2.1/csd01/eox-core-v1.0-csd01.html
https://docs.oasis-open.org/openeox/eox-core/v2.1/csd01/eox-core-v1.0-csd01.pdf

Previous stage:

N/A

Latest stage:

https://docs.oasis-open.org/csaf/csaf/v2.1/csaf-v2.1.md (Authoritative)
https://docs.oasis-open.org/openeox/eox-core/v1.0/eox-core-v1.0.html
https://docs.oasis-open.org/openeox/eox-core/v1.0/eox-core-v1.0.pdf

Technical Committee:

OASIS OpenEOX TC

Chair:

Justin Murphy (justin.murphy@cisa.dhs.gov), DHS Cybersecurity and Infrastructure Security Agency (CISA)
Omar Santos (osantos@cisco.com), Cisco Systems

Editors:

Jautau White (jaywhite@microsoft.com), Microsoft Corporation
Stefan Hagen (stefan@hagen.link), Individual
Thomas Schmidt (thomas.schmidt@bsi.bund.de), Federal Office for Information Security (BSI) Germany

Additional artifacts:

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

- EOX Core JSON schema: https://docs.oasis-open.org/openeox/eox-core/v1.0/csd01/schema/eox-core.json.
  Latest stage: https://docs.oasis-open.org/openeox/eox-core/v1.0/schema/eox-core.json.

Related work:

This specification replaces or supersedes:

n/a

Declared JSON namespaces:

- https://docs.oasis-open.org/openeox/eox-core/v1.0/schema/eox-core.json

Abstract:

The OpenEoX Core Schema defines the core schema for the OpenEOX unified machine-readable approach to managing and sharing End-of-Life (EOL) and End-of-Support (EOS) information for commercial and open source software and hardware.

Status:

Key words:

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

Citation format:

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

[EOX-Core-v1.0]

OpenEOX Core Schema Version 1.0. Edited by Jautau White, Stefan Hagen, and Thomas Schmidt. 21 January 2026. OASIS Committee Specification Draft 01. https://docs.oasis-open.org/openeox/eox-core/v1.0/csd01/eox-core-v1.0-csd01.htmll. Latest stage: https://docs.oasis-open.org/openeox/eox-core/v1.0/eox-core-v1.0-csd01.html.

Notices

Table of Contents

1 Introduction

1.1 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:

Example 1:

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

All examples in this document are informative only.

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

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

2 Overview

TBD

3 Design Considerations

The OpenEoX provides the vocabulary for exchanging standardized End-of-X (EoX) life cycle data for any product. This includes, but is not limited to hardware, software and specifications, as well as all supplier-consumer relationships, services and offerings.

3.1 OpenEoX Architecture

The OpenEoX standard is architected as a modular three-layer framework:

3.1.1 OpenEoX Core (this specification)

The OpenEoX Core specification contains the fundamental EoX life cycle information elements and is designed to be imported by other standards or integrated directly into product responses. The Core schema establishes one part of the essential data elements required for any EoX statement:

Required fields:

- `end_of_life`: The definitive timestamp when all vendor support ceases
- `end_of_security_support`: The critical timestamp when security remediation commitments end
- `last_updated`: The timestamp tracking for data freshness and change management

Optionally, the standard also conveys additional life cycle events.

The core information can be provided independently when the product context is available through other means, such as SNMP responses, HTTPS API responses.

3.1.2 OpenEoX Shell

The OpenEoX Shell specification provides the binding mechanism that combines OpenEoX Core life cycle information to specific product identifications. This combination is called an “OpenEoX statement” - a complete, standalone representation of life cycle information for a specific product. The Shell forms the standalone mode in OpenEoX, enabling complete lifecycle statements that can be processed independently without external context.

3.1.3 OpenEoX API

The OpenEoX API specification addresses the distribution, discovery, and consumption of OpenEoX Core and Shell data across vendor ecosystems and consumer applications. The API specification defines standardized endpoints, query mechanisms, and data exchange patterns that enable automated integration.

3.2 Design Principles

3.2.1 Standardization and Interoperability

OpenEoX addresses the current industry challenge where vendors use proprietary, inconsistent formats for communicating life cycle information. By establishing a common JSON-based format, OpenEoX enables:

- Automated processing and integration across vendor ecosystems
- Consistent interpretation of lifecycle stages regardless of vendor
- Reduced operational overhead in managing multi-vendor environments
- Enhanced accuracy in infrastructure lifecycle planning

3.2.2 Security-First Approach

Recognition that End-of-Security-Support represents a critical inflection point for infrastructure security has influenced the OpenEoX design. The mandatory `end_of_security_support` field ensures that security life cycle information is always available, supporting:

- Proactive vulnerability management planning
- Compliance with security frameworks requiring current support status
- Risk assessment for critical infrastructure components
- Automated alerting when security support approaches termination

3.3 Industry Applications

3.3.1 Vendor Perspectives

OpenEoX enables vendors to:

- Standardize EoX communication across all product lines and channels
- Reduce support inquiries related to life cycle confusion
- Demonstrate proactive customer stewardship through clear life cycle communication
- Support automated customer notification systems
- Facilitate integration with partner and reseller systems

3.3.2 Consumer Perspectives

Organizations consuming OpenEoX information benefit through:

- Automated discovery of life cycle status across multi-vendor environments
- Proactive planning for product refreshes and migrations
- Enhanced compliance reporting for security and regulatory frameworks
- Reduced operational risk from unexpected support terminations
- Streamlined asset management and inventory tracking

The OpenEoX architecture ensures that organizations can adopt OpenEoX in a manner that aligns with their existing infrastructure and operational models while preserving the ability to enhance integration over time.

3.4 Construction Principles

The OpenEoX Core specification describes a map for well-known named transition events (like end-of-life) to dates. Producers use this mapping to bind products to this minimal set of life cycle events.

Other schemata provide models to add the data needed for product identification. Examples for such schemata are:

- OpenEoX Shell
- OpenEoX API
- integration into the product itself (for example via SNMP calls or HTTPS requests)
- integration into other standards (for example CSAF)

This separation provides consistent, interoperable, actionable, structured, and validated life cycle information with the goal to offer complete automation.

The chosen data format and version “JSON Schema Draft 2020-12” [JSON-Schema-Core] allows validation and delegation to providers of referenced schemata. The latter enables separation of concerns as it allows other standards, specifications, and product-specific implementations to import and reference the OpenEoX Core schema. This enables consistency and interoperability across different implementations.

The OpenEoX Core schema is defined at https://docs.oasis-open.org/openeox/eox-core/v1.0/schema/eox-core.json

The schema elements and their expected usage patterns are detailed in subsequent sections.

Linking and integration of the OpenEoX Core schema with product identification schema information may lead to inconsistencies. The producers of OpenEoX information items have to ensure the intended semantics. For example, that lifecycle dates are ordered logically. Semantic validation is out of scope for this OpenEoX Core specification.

The OpenEoX Core schema is designed to be self-contained with minimal external dependencies, utilizing only:

[RFC3339] Date and time format specification for timestamp representation
[JSON-Schema-Core] Schema validation and structure definition

OpenEoX information items SHOULD only contain members where the value is an OpenEoX life cycle date. The use of placeholders or `null` values is discouraged and any compliant OpenEoX implementation SHOULD ignore such members.

> In combination, these to preceding requirements maximize clarity and simplify automation.

OpenEoX information items SHOULD NOT contain additional properties in themselves or as part of referenced schema instances. Suggestions for new fields or values SHOULD be made through issues in the TC's GitHub. The JSON schemas defined in this standard do not allow the use of additional properties and custom keywords.

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

3.5 Format Validation

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

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://docs.oasis-open.org/openeox/eox-core/v1.0/schema/meta.json",
  "$dynamicAnchor": "meta",
  "$vocabulary": {
    "https://json-schema.org/draft/2020-12/vocab/core": true,
    "https://json-schema.org/draft/2020-12/vocab/format-assertion": true
  },
  "allOf": [
    { "$ref": "https://json-schema.org/draft/2020-12/meta/core" },
    { "$ref": "https://json-schema.org/draft/2020-12/meta/format-assertion" }
  ]
}
```

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

```
{
  "$schema": "https://docs.oasis-open.org/openeox/eox-core/v1.0/schema/meta.json",
  // ...
}
```

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

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

3.6 Date and Time

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

- The letter `T` separating the date and time SHALL be upper case.
- The separator between date and time MUST be the letter `T`.
- The letter `Z` indicating the timezone UTC SHALL be upper case.
- Fractions of seconds are allowed as specified in the standards mention above with the full stop (`.`) as separator.
- Empty timezones MUST NOT be used.
- The ABNF of [RFC3339], section 5.6 applies.

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

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

4 Taxonomy

The following subsections describe the taxonomy defining and explaining all terms the standard is build upon.

4.1 End-of-Life

The End-of-Life (EoL) indicates the last day when the particular product (or the product version/release) is officially supported in any way by the vendor. After this date there shouldn't be more development, updates or any type of support expected from the vendor. The existing customers are also impacted by this product lifecycle stage and should consider migration to the still supported product version or release.

4.2 End-of-Sales

The End-of-Sales (EoS) indicates the last day when a particular product (or the product version/release) may be ordered by customers from vendor sales channels. After this date, the product is no longer for sale. However, there might be other sources where the product is still available. Once the product reaches the EoS lifecycle stage, it may still be supported by the vendor, based on the official or dedicated vendor lifecycle model for this product, for existing customers. The implications for existing customers regarding license renewals, updates, upgrades to newer versions or ongoing technical support can vary depending on the vendor's specific policies.

4.3 End-of-Security-Support

The End-of-Security-Support (EoSSec) indicates the last day when the vendor has committed to providing security remediations for the particular product (or the product version/release).

4.4 Product

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

4.5 Product Lifecycle

Every product type (software, hardware, managed service and other deliverables) can be associated with a lifecycle model. It can contain definitions of various support models (different levels of maintenance) in association to the product versioning convention. The lifecycle support model can be dynamic and may change over time, from the product's initial release (General Availability) to its discontinuation (End-of-Life). During the product lifecycle, support models may switch from one state to another and may even run in parallel to meet individual requirements. Those requirements may depend on the product type, the vendor offerings, as well as geographical related regulations.

4.6 Vendor

Vendor refers to the community, individual, or organization that created or maintains a product (software, hardware, managed service and other deliverables). This includes, but is not limited to, open-source software and hardware providers.

5 Schema Elements

The OpenEoX Core schema describes how to represent OpenEoX core information as a JSON document.

The OpenEoX Core schema Version 1.0 builds on the JSON Schema draft 2020-12 rules extended by the format validation enforcement (see 3.5).

```
  "$schema": "https://docs.oasis-open.org/openeox/eox-core/v1.0/schema/meta.json",
```

The schema identifier is:

```
  "$id": "https://docs.oasis-open.org/openeox/v1.0/schema/core.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 OpenEoX core information

Types and properties together provide the vocabulary for the domain specific language supporting OpenEoX information.

The four mandatory properties are `$schema`, `end_of_life`, `end_of_security_support`, and `last_updated`. The additional property, `end_of_sales`, is optional.

5.1 Definitions

The definitions (`$defs`) introduce the following domain specific types into the OpenEoX language: EoX Timestamp (`eox_timestamp_t`).

```
  "$defs": {
    "eox_timestamp_t": {
      // ...
    }
  },
```

5.1.1 EoX Timestamp Type

The EoX Timestamp (`eox_timestamp_t`) type of value type `string` contains the timestamp at which the product reaches the specified stage or the indicator that this is still ‘to be announced’.

```
    "oneOf": [
      {
        "format": "date-time"
      },
      {
        "const": "tba"
      }
    ]
```

Therefore, it MUST either be a `string` with format `date-time` or use the constant value `tba`.

OpenEoX Consumer MUST treat `tba` as a date greater than the maximum date. In any semantic interpretation, the value `tba` MUST be treated as ‘not yet announced’.

5.2 Properties

The following subsections document the five properties of a OpenEoX Core JSON. The four mandatory properties are `$schema`, `end_of_life`, `end_of_security_support`, and `last_updated`. The property `end_of_sales` is optional.

5.2.1 Schema Property

The OpenEoX Core Schema (`$schema`) of value type `string` and `const` specifies the schema the JSON object must be valid against. The single valid value for this `const` is:

```
  https://docs.oasis-open.org/openeox/v1.0/schema/core.json
```

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

5.2.2 End-of-Life Property

End-of-Life (`end_of_life`) of value type `eox_timestamp_t` indicates the last day when the particular product is officially supported in any way by the vendor.

5.2.3 End-of-Sales Property

End-of-Sales (`end_of_sales`) of value type `eox_timestamp_t` indicates the last day when a particular product may be ordered by customers from vendor sales channels.

5.2.4 End-of-Security-Support Property

End-of-Security-Support (`end_of_security_support`) of value type `eox_timestamp_t` indicates the last day when the vendor has committed to providing security remediations for the particular product.

5.2.5 Last Updated Property

Timestamp of last change (`last_updated`) of value type `string` with format `date-time` contains the RFC 3339 timestamp when the record was last updated.

6 Tests

The following 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 OpenEoX Core Information. A program MUST handle a test failure as an error.

6.1.1 Inconsistent EoL Date

It MUST be tested that the `end_of_life` is later than or equal to any other date value in the OpenEoX Core Information. The property `last_updated` is ignored in this test. As the timestamps might use different timezones, the sorting MUST take timezones into account. Except for `end_of_life`, any property of type `eox_timestamp_t` with value `tba` MUST be ignored in the comparison.

The relevant path for this test is:

```
  /end_of_life
```

Example 1 (which fails the test):

```
{
  // ...
  "end_of_life": "2025-11-19T16:00:00Z",
  "end_of_security_support": "2025-12-31T12:00:00Z",
  // ...
}
```

The `end_of_security_support` is later than the `end_of_life`.

6.1.2 Date and Time

For each item of type `string` and format `date-time` it MUST be tested that it conforms to the rules given in section 3.6. Any property with the value `tba` is ignored in this comparison.

The relevant path for this test is:

```
/end_of_life
/end_of_sales
/end_of_security_support
/last_updated
```

Example 1 (which fails the test):

```
"end_of_life": "2025-12-31 12:00:00Z",
```

The `end_of_life` uses a white space as separator instead the letter `T`.

6.2 Recommended Tests

Recommended tests SHOULD NOT fail at a valid OpenEoX Core Information without a good reason. Failing such a test does not make the OpenEoX Core Information invalid. These tests may include information about features which are still supported but expected to be deprecated in a future version of OpenEoX. A program MUST handle a test failure as a warning.

6.2.1 Use of `tba` where Date is set for EoL

For each property of type `eox_timestamp_t`, it MUST be tested that it does not contain the value `tba` if the `end_of_life` value is set to a date.

The relevant path for this test is:

```
/end_of_sales
/end_of_security_support
```

Example 1 (which fails the test):

```
{
  // ...
  "end_of_life": "2025-09-01T12:00:00Z",
  "end_of_security_support": "tba",
  // ...
}
```

The `end_of_security_support` is set to `tba` but `end_of_life` already has a date.

A tool MAY set the `end_of_security_support` to the same value as `end_of_life` as a quick fix.

6.3 Informative Tests

Informative tests provide insights in common mistakes and bad practices. They MAY fail at a valid OpenEoX Core Information. 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 EoSSec before EoS

It MUST be tested that if `end_of_sales` is present, its value is before or equal to the value of `end_of_security_support`. As the timestamps might use different timezones, the sorting MUST take timezones into account. The test MUST be skipped, if `end_of_sales` is set to `tba`.

The relevant path for this test is:

```
  /end_of_sales
```

Example 1 (which fails the test):

```
{
  // ...
  "end_of_sales": "2025-12-31T12:00:00Z",
  "end_of_security_support": "2025-11-19T16:00:00Z",
  // ...
}
```

The `end_of_security_support` is earlier than the `end_of_sales`.

7 Conformance

The single subsection of this section lists the conformance targets and clauses. The clauses directly corresponding to the targets are listed in separate sub-subsections of the target list subsection.

The order in which targets, and their corresponding clauses appear is somewhat arbitrary as there is no natural order on such diverse roles participating in the EoX exchanging ecosystem.

7.1 Conformance Targets

This document defines requirements for the file format and for certain software components that interact with it. The entities ("conformance targets") for which this document defines requirements are:

- **OpenEoX Core Information**: An EoX information in the format defined by this document.
- **OpenEoX Core Producer**: A program which emits output in the OpenEoX Core format.
- **OpenEoX Core Consumer**: A program that reads and interprets OpenEoX Core Information.
- **OpenEoX Core Basic Validator**: A program that reads a JSON object and checks it against the JSON schema and performs mandatory tests.
- **OpenEoX Core Extended Validator**: A OpenEoX Core Basic Validator that additionally performs recommended tests.
- **OpenEoX Core Full Validator**: A OpenEoX Core Extended Validator that additionally performs guidance tests.
- **OpenEoX Core Library**: A library that implements OpenEoX Core data capabilities.
- **OpenEoX Core Library with Basic Validation**: A OpenEoX Core Library that also satisfies the conformance target "OpenEoX Core Basic Validator".
- **OpenEoX Core Library with Extended Validation**: A OpenEoX Core Library that also satisfies the conformance target "OpenEoX Core Extended Validator".
- **OpenEoX Core Library with Full Validation**: A OpenEoX Core Library that also satisfies the conformance target "OpenEoX Core Full Validator".
- **OpenEoX Core Differ**: A program that compares OpenEoX Core Information and calculates the changes.
- **OpenEoX Core Comparator**: A program that compares OpenEoX Core Information and returns the newer one.
- **OpenEoX Core Sorter**: A program that sorts OpenEoX Core Information from newest to oldest or vice versa.
- **OpenEoX Core Merger**: A program that combines OpenEoX Core Information.

7.1.1 Conformance Clause 1: OpenEoX Core Information

A text file or data stream satisfies the "OpenEoX Core Information" conformance profile if it:

- conforms to the syntax and semantics defined in section 3.5.
- conforms to the syntax and semantics defined in section 3.6.
- conforms to the syntax and semantics defined in section 5.
- does not fail any mandatory test defined in section 6.1.

7.1.2 Conformance Clause 2: OpenEoX Core Producer

A program satisfies the "OpenEoX Core Producer" conformance profile if the program:

- produces output in the OpenEoX Core format, according to the conformance profile "OpenEoX Core Information".
- satisfies those normative requirements in section 5 and Appendix B that are designated as applying to OpenEoX Producers.

7.1.3 Conformance Clause 3: OpenEoX Core Consumer

A processor satisfies the “OpenEoX Core Consumer” conformance profile if the processor:

- reads OpenEoX Core Information and interprets them according to the semantics defined in section 5.
- satisfies those normative requirements in section 5 and Appendix B that are designated as applying to OpenEoX Consumers.

7.1.4 Conformance Clause 4: OpenEoX Core Basic Validator

A program satisfies the “OpenEoX Core Basic Validator” conformance profile if the program:

- reads JSON objects and performs a check against the JSON schema.
- performs all mandatory tests as given in section 6.1.
- does not change the OpenEoX Core Information.

A OpenEoX Core Basic Validator MAY provide one or more additional functions:

- Only run one or more selected mandatory tests.
- Apply quick fixes as specified in the standard.
- Apply additional quick fixes as implemented by the vendor.

7.1.5 Conformance Clause 5: OpenEoX Core Extended Validator

A OpenEoX Core Basic Validator satisfies the “OpenEoX Core Extended Validator” conformance profile if the OpenEoX Core Basic Validator:

- satisfies the “OpenEoX Core Basic Validator” conformance profile.
- additionally performs all recommended tests as given in section 6.2.
- provides a function that allows to specify tests for which test results of warning are reported as error.

A OpenEoX Core Extended Validator MAY provide an additional function to only run one or more selected recommended tests.

7.1.6 Conformance Clause 6: OpenEoX Core Full Validator

A OpenEoX Core Extended Validator satisfies the “OpenEoX Core Full Validator” conformance profile if the OpenEoX Core Extended Validator:

- satisfies the “OpenEoX Core Extended Validator” conformance profile.
- additionally performs all guidance tests as given in section 6.3.
- provides a function that allows to specify tests for which test results of information are reported as warning.
- provides a function that allows to specify tests for which test results of information are reported as error.

A OpenEoX Core Full Validator MAY provide an additional function to only run one or more selected guidance tests.

7.1.7 Conformance Clause 7: OpenEoX Core Library

A library satisfies the “OpenEoX Core Library” conformance profile if the library:

- implements all elements as data structures conforming to the syntax and semantics defined in sections 3.5, 3.6 and 5.
- checks all elements according to the patterns provided in the JSON schema.
- has a function that reads OpenEoX Core Information into the data structure from a
  - file system.
  - URL.
  - data stream.
- provides function for sorting the keys and sorts the keys automatically on output.
- has a function that outputs the data structure as OpenEoX Core Information
  - on the file system.
  - as string.
  - into a data stream.

The library MAY implement an option to retrieve the keys unsorted.

7.1.8 Conformance Clause 8: OpenEoX Core Library with Basic Validation-1

A OpenEoX Core Library satisfies the “OpenEoX Core Library with Basic Validation” conformance profile if the OpenEoX Core Library:

- satisfies the “OpenEoX Core Library” conformance profile.
- satisfies the “OpenEoX Core Basic Validator” conformance profile.
- validates the OpenEoX Core Information before output according to the “OpenEoX Core Basic Validator” and presents the validation result accordingly.
- provide a function to validate the data structure in its current state according to the “OpenEoX Core Basic Validator” and presents the validation result accordingly.

A OpenEoX Core Library does not satisfies the “OpenEoX Core Library with Basic Validation” conformance profile if the OpenEoX Core Library uses an external library or program for the “OpenEoX Core Basic Validator” part and does not enforce its presence.

7.1.9 Conformance Clause 9: OpenEoX Core Library with Extended Validation

A OpenEoX Core Library satisfies the “OpenEoX Core Library with Extended Validation” conformance profile if the OpenEoX Core Library:

- satisfies the “OpenEoX Core Library” conformance profile.
- satisfies the “OpenEoX Core Extended Validator” conformance profile.
- validates the OpenEoX Core Information before output according to the “OpenEoX Core Extended Validator” and presents the validation result accordingly.
- provide a function to validate the data structure in its current state according to the “OpenEoX Core Extended Validator” and presents the validation result accordingly.

A OpenEoX Core Library does not satisfies the “OpenEoX Core Library with Extended Validation” conformance profile if the OpenEoX Core Library uses an external library or program for the “OpenEoX Core Extended Validator” part and does not enforce its presence.

7.1.10 Conformance Clause 10: OpenEoX Core Library with Full Validation

A OpenEoX Core Library satisfies the “OpenEoX Core Library with Extended Validation” conformance profile if the OpenEoX Core Library:

- satisfies the “OpenEoX Core Library” conformance profile.
- satisfies the “OpenEoX Core Full Validator” conformance profile.
- validates the OpenEoX Core Information before output according to the “OpenEoX Core Full Validator” and presents the validation result accordingly.
- provide a function to validate the data structure in its current state according to the “OpenEoX Core Full Validator” and presents the validation result accordingly.

A OpenEoX Core Library does not satisfies the “OpenEoX Core Library with Full Validation” conformance profile if the OpenEoX Core Library uses an external library or program for the “OpenEoX Core Full Validator” part and does not enforce its presence.

7.1.11 Conformance Clause 11: OpenEoX Core Differ

A program satisfies the “OpenEoX Core Differ” conformance profile if the program:

- satisfies the “OpenEoX Core Consumer” conformance profile.
- calculates the difference between two OpenEoX Core Information JSON Objects and presents it as JSON Diff.

A OpenEoX Core Differ MAY choose to make that information also available in other data formats.

7.1.12 Conformance Clause 12: OpenEoX Core Comparator

A program satisfies the “OpenEoX Core Comparator” conformance profile if the program:

- satisfies the “OpenEoX Core Consumer” conformance profile.
- provides a function to compares two OpenEoX Core Information Objects `a` and `b` based on their `last_updated` value and returns:
  - `-1` if `a["last_updated"] < b["last_updated"]`
  - `0` if `a["last_updated"] = b["last_updated"]`
  - `1` if `a["last_updated"] > b["last_updated"]`
- provides a function to compares two OpenEoX Core Information Objects `a` and `b` based on their `last_updated` value and returns:
  - `b` if `a["last_updated"] < b["last_updated"]`
  - `a` if `a["last_updated"] = b["last_updated"]`
  - `a` if `a["last_updated"] > b["last_updated"]`
- takes timezones into account.

A OpenEoX Core Comparator MAY choose to make that information also available in other data formats.

7.1.13 Conformance Clause 13: OpenEoX Core Sorter

A program satisfies the "OpenEoX Core Sorter" conformance profile if the program:

- satisfies the "OpenEoX Core Consumer" conformance profile.
- provides a function to sort an arbitrary number of OpenEoX Core Information Objects `last_updated` value with that value descending (default).
- provides a function to sort an arbitrary number of OpenEoX Core Information Objects `last_updated` value with that value ascending.
- takes timezones into account.

A OpenEoX Core Sorter MAY choose to make that information also available in other data formats.

7.1.14 Conformance Clause 14: OpenEoX Core Merger

A program satisfies the "OpenEoX Core Merger" conformance profile if the program:

- satisfies the "OpenEoX Core Consumer" conformance profile.
- provides a function to merge OpenEoX Core Information into a single OpenEoX Core Information Object by overwriting all old values of keys present in newer OpenEoX Core Information Objects with the values from newer objects.

  > This includes the value of `last_updated`.

- provides a function to merge OpenEoX Core Information into a single OpenEoX Core Information Object by overwriting all old values of keys present in newer OpenEoX Core Information Objects with the values from newer objects but treating `tba` as not present during the overwrite process.

  > This includes the value of `last_updated`. Ignoring `tba` while overwriting ensures that a date value is never overwritten by `tba`.

- provides a function to merge OpenEoX Core Information into a single OpenEoX Core Information Object by taking the newest as base and iterative appending keys that are not present in that object while iterating over the OpenEoX Core Information from newest to oldest.

  > As `last_updated` is a required field, it is not changed during the process. Values of keys that are inserted are never changed in that process.

- takes timezones into account.

A program MAY implement other algorithms than defined here, if and only if, the results are guaranteed to be the same.

A OpenEoX Core Merger MAY choose to make that information also available in other data formats.

Appendix A. References

[This appendix contains the normative and informative references that are used in this document.]

[While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.]

A.1 Normative References

[The following documents are referenced in such a way that some or all of their content constitutes requirements of this document.]

[https://262.ecma-international.org/15.0/]

[https://www.iso.org/standard/70907.html.]

[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.

[https://datatracker.ietf.org/doc/html/draft-bhutton-relative-json-pointer-00.]

[https://www.rfc-editor.org/info/rfc2119.]

[https://www.rfc-editor.org/info/rfc3339.]

[https://www.rfc-editor.org/info/rfc4180.]

[https://www.rfc-editor.org/info/rfc7230.]

[https://www.rfc-editor.org/info/rfc7464.]

[https://www.rfc-editor.org/info/rfc8174.]

[https://www.rfc-editor.org/info/rfc8259.]

[https://www.rfc-editor.org/info/rfc9562.]

[https://spdx.github.io/spdx-spec/.]

A.2 Informative References

[TBD]

Appendix B. Safety, Security and Data Protection

All safety, security, and data protection requirements relevant to the context in which OpenEoX information items are used MUST be translated into, and consistently enforced through, OpenEoX implementations and processes.

OpenEoX information items are based on JSON, thus the security considerations of [RFC8259] apply and are repeated here as service for the reader:

> Generally, there are security issues with scripting languages. JSON is a subset of JavaScript but excludes assignment and invocation.
> Since JSON's syntax is borrowed from JavaScript, it is possible to use that language's `eval()` function to parse most JSON texts (but not all; certain characters such as `U+2028 LINE SEPARATOR` and `U+2029 PARAGRAPH SEPARATOR` are legal in JSON but not JavaScript). This generally constitutes an unacceptable security risk, since the text could contain executable code along with data declarations. The same consideration applies to the use of eval()-like functions in any other programming language in which JSON texts conform to that language's syntax.

Appendix C. Acknowledgments

The following individuals were members of the OASIS OpenEOX Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

OASIS OpenEOX TC Members:

First Name	Last Name	Company
Denny	Page	TIBCO Software Inc.
Feng	Cao	Oracle
Jautau	White	Microsoft Corporation
Justin	Murphy	DHS Cybersecurity and Infrastructure Security Agency(CISA)
Langley	Rock	Red Hat
Martin	Prpic	Red Hat
Omar	Santos	Cisco Systems
Stefan	Hagen	Individual
Thomas	Schmidt	Federal Office for Information Security (BSI) Germany
…	…	…

Appendix D. Revision History

Revision	Date	Editor	Changes Made
eox-core-v1.0-wd20250716-dev	2025-07-16	Jautau White, Stefan Hagen, and Thomas Schmidt	Preparing initial Editor Revision
eox-core-v1.0-wd20250820-dev	2025-08-20	Jautau White, Stefan Hagen, and Thomas Schmidt	Editor Revision with first structure
eox-core-v1.0-wd20251119-dev	2025-11-19	Jautau White, Stefan Hagen, and Thomas Schmidt	Editor Revision for TC review
eox-core-v1.0-wd20260121-dev	2026-01-21	Jautau White, Stefan Hagen, and Thomas Schmidt	Editor Revision for TC review

Appendix E. Notices