GCVE-BCP-10 - Improved Common Platform Enumeration for GCVE

GCVE.eu

GCVE.eu

Contents

1 GCVE-BCP-10: Improved Common Platform Enumeration for GCVE

- **Version**: 1.1
- **Status**: Draft (for Public Review)
- **Date**: 2026-04-29
- **Authors**: GCVE Working Group
- **BCP ID**: BCP-10

1.1 Abstract

This document specifies an improved platform enumeration model for GCVE aligned with the current implementation of `cpe-editor`.

The model remains compatible with existing Common Platform Enumeration (CPE) practices and string formats, while adding registry records for vendors, products, CPE entries, metadata, relationships, and optional moderation proposals.

The main improvements are:

- deterministic UUIDv5 identifiers for vendor and product records;
- a documented UUID namespace hierarchy for repeatable imports across instances;
- first-class vendor, product, and CPE entry records matching the `cpe-editor` export format;
- explicit metadata using namespaced metadata keys such as `gcve:url` and `gcve:description`;
- support for synonym, rename, equivalence, and lifecycle relationships through structured relationship records;
- audit-oriented timestamps and approval fields for metadata and relationship records;
- registry support for vendor, product, CPE, metadata, and relationship synchronization.

The goal is to preserve interoperability with existing CPE-based tooling while addressing operational limitations encountered in long-term platform naming, vendor consolidation, product renaming, alias management, and decentralized registry synchronization.

1.2 Motivation

Traditional CPE naming is useful as a compact and widely understood naming format, but it has several practical limitations:

- the name itself often acts as the identifier;
- renaming or normalization can break references;
- aliases and synonyms are difficult to track cleanly;
- provenance and approval state are not always explicit;
- vendor mergers, splits, rebranding, and product renaming require registry-level lifecycle handling;
- non-matching metadata such as URLs, descriptions, notes, and external references is difficult to attach consistently;
- repeated imports from public CPE sources need deterministic identity generation to remain idempotent across registry instances.

GCVE-BCP-10 addresses these limitations by separating the **stable identity** of a registry record from its **displayed name**, **CPE-compatible representation**, and **descriptive metadata**.

1.3 Design principles

GCVE-BCP-10 follows these principles:

- **Backward compatibility:** existing CPE 2.3 strings remain valid and central to matching.
- **Stable identity:** UUIDs are the primary durable identifiers for vendor and product records.
- **Deterministic import:** vendor and product UUIDs are generated with UUIDv5 from normalized names.
- **Vendor-scoped product identity:** product UUIDs are derived from both the normalized vendor name and normalized product name.
- **Rename safety:** names and titles may change without changing the identity of the record.
- **Namespaced metadata:** metadata keys use namespace prefixes, with GCVE-defined keys using the `gcve:` prefix.
- **Structured synonym handling:** aliases, equivalent names, and renames are modeled as first-class relationship records.

- **Auditability:** submitted, approved, created, and updated timestamps are preserved where applicable.
- **Registry governance:** GCVE operates a registry for updates, synonym resolution, metadata approval, and vendor/product management.

1.4 Terminology

1.4.1 Vendor record

A structured record describing a vendor or organization. A vendor record has a stable UUID, a normalized machine-oriented name, a human-readable title, optional notes, and timestamps.

1.4.2 Product record

A structured record describing a product, platform, application, operating system, hardware item, or other CPE-relevant entity. Product records use stable UUIDs and are linked to vendor records by `vendor_uuid`.

1.4.3 CPE entry record

A structured CPE 2.3 entry linked to a vendor and product by UUID. A CPE entry stores the full CPE URI, parsed CPE attributes, the optional upstream `cpeNameId`, deprecation status, notes, and timestamps.

1.4.4 Metadata record

A structured key/value assertion attached to a vendor or product record. Metadata keys are namespaced. GCVE-defined metadata keys use the `gcve.` prefix.

1.4.5 Canonical name

The preferred name or CPE-compatible string currently designated by the GCVE registry for a vendor or product record. Canonical naming may be represented through normalized record fields, metadata, and relationships.

1.4.6 Alias / synonym

An alternative vendor or product name that refers to the same or closely related identity.

1.4.7 Relationship record

A typed link between a source vendor or product UUID and a target vendor or product UUID.

1.4.8 CPE-compatible string

A CPE 2.3 formatted string or compatible match criterion used by existing vulnerability and asset-management tooling.

1.5 Alignment with `cpe-editor`

The current `cpe-editor` implementation exports a dataset using the following top-level structure:

```
{
  "format": "cpe-editor-dataset",
  "version": "1",
  "exported_at": "2026-04-26T17:00:00+00:00",
  "counts": {
    "vendors": 2,
    "products": 1,
    "cpes": 1,
    "metadata": 2,
    "relationships": 1,
    "proposals": 0
  },
  "vendors": [],
  "products": [],
  "cpes": [],
  "metadata": [],
  "relationships": [],
  "proposals": []
}
```

GCVE-BCP-10 therefore treats the `cpe-editor` dataset archive JSON as the reference exchange shape for this draft. In particular:

- `vendors` and `products` carry deterministic UUIDv5 identifiers;
- `cpes` carry CPE 2.3 strings and parsed CPE fields;

- `metadata` carries namespaced key/value assertions for vendors and products;
- `relationships` carries approved or imported synonym, rename, and lifecycle links;
- `proposals` is an implementation-specific moderation-history array and may be empty.

1.6 UUID generation

1.6.1 UUID format

GCVE-BCP-10 uses standard lowercase textual UUIDs in canonical 8-4-4-4-12 form:

```
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

Vendor and product UUIDs generated by `cpe-editor` are UUIDv5 identifiers. UUIDv5 is name-based and deterministic: the same namespace UUID and same normalized name string produce the same UUID.

1.6.2 Namespace hierarchy

The current `cpe-editor` namespace hierarchy is:

```python
from uuid import NAMESPACE_URL, uuid5

GCVE_ROOT_NAMESPACE_URL = "GCVE-BCP-10"
GCVE_ROOT_NAMESPACE = uuid5(NAMESPACE_URL, GCVE_ROOT_NAMESPACE_URL)

VENDOR_UUID_NAMESPACE = uuid5(GCVE_ROOT_NAMESPACE, "vendor")
PRODUCT_UUID_NAMESPACE = uuid5(GCVE_ROOT_NAMESPACE, "product")
```

This produces the following namespace UUIDs:

Name	UUID
`GCVE_ROOT_NAMESPACE`	`12f50db3-9dac-5340-843f-1a9823090227`
`VENDOR_UUID_NAMESPACE`	`bc564076-4070-5fd8-8b46-27679548dd7a`
`PRODUCT_UUID_NAMESPACE`	`d1b407e8-4bd6-5fda-ad3b-21d12be51c07`

1.6.3 Token normalization

Before UUID generation, vendor and product tokens are normalized as follows:

```python
def normalize_token(value: str) -> str:
    return (value or "").strip().lower().replace(" ", "_")
```

This means:

- leading and trailing whitespace is removed;
- text is converted to lowercase;
- ASCII space characters are replaced with underscores;
- hyphens are preserved;
- existing underscores are preserved;
- no other CPE escaping or Unicode normalization is implied by this profile.

1.6.4 Vendor UUID generation

A vendor UUID is generated from the normalized vendor name:

```python
def vendor_uuid_for_name(name: str) -> str:
    return str(uuid5(VENDOR_UUID_NAMESPACE, normalize_token(name)))
```

Examples:

Vendor name input	Normalized name	UUIDv5
misp	misp	536016cd-a314-5880-947a-e465002d0fab
misp-project	misp-project	c759b712-4932-513a-b6a1-f86c0bcc1a5e
MISP Project	misp_project	932a354b-26eb-5651-ae79-eba7f4658ba2

1.6.5 Product UUID generation

A product UUID is generated from the normalized vendor name and normalized product name, separated by a colon:

```python
def product_uuid_for_names(vendor_name: str, product_name: str) -> str:
    return str(
        uuid5(
            PRODUCT_UUID_NAMESPACE,
```

```
            f"{normalize_token(vendor_name)}:{normalize_token(
                product_name)}",
        )
    )
```

The product UUID is vendor-scoped. This avoids collisions where different vendors legitimately use the same product token.

Examples:

Vendor input	Product input	UUIDv5 name string	UUIDv5
`misp`	`misp`	`misp:misp`	`f28e0890-4e1b-59a5-b52c-3e9354071db7`
`misp-project`	`misp`	`misp-project:misp`	`e75bc4d3-b693-577b-ac4b-40d13b5f5bc5`

1.6.6 UUID stability rules

- A vendor UUID MUST be generated with `uuid5(VENDOR_UUID_NAMESPACE, normalize_token(vendor_name))` when importing or backfilling vendor records.
- A product UUID MUST be generated with `uuid5(PRODUCT_UUID_NAMESPACE, normalize_token(vendor_name)+ ":"+ normalize_token(product_name))` when importing or backfilling product records.
- A product UUID MUST be scoped to its vendor name as shown above.
- Vendor and product UUIDs MUST remain stable across repeated imports of the same normalized names.
- A change in `title`, `notes`, metadata, or CPE entries MUST NOT change the UUID.
- If a registry changes the machine-oriented `name`, it MUST either preserve the existing UUID through registry policy or create a new deterministic UUID and add a relationship such as `renamed-to`, `synonym-of`, or `superseded-by`.
- Random UUIDs MAY be used temporarily for records created interactively before normalization is final, but exported vendor and product records SHOULD be backfilled to the deterministic UUIDv5 values.

1.7 Data model

GCVE-BCP-10 uses separate registry records for vendors, products, CPE entries, metadata, and relationships.

1.7.1 Vendor record

A vendor record has the following fields:

- `uuid`: stable UUID for the vendor.
- `name`: normalized machine-oriented vendor name.
- `title`: human-readable vendor title; may be **`null`**.
- `notes`: optional registry notes; may be **`null`**.
- `created_at`: creation timestamp.
- `updated_at`: last update timestamp.

Example:

```
{
  "created_at": "2026-04-26T14:25:34.846011",
  "name": "misp",
  "notes": null,
  "title": "Misp",
  "updated_at": "2026-04-26T14:25:34.846011",
  "uuid": "536016cd-a314-5880-947a-e465002d0fab"
}
```

Another vendor record for an alias or related organization:

```
{
  "created_at": "2026-04-26T14:24:53.155776",
  "name": "misp-project",
  "notes": null,
  "title": "Misp Project",
  "updated_at": "2026-04-26T14:24:53.155776",
  "uuid": "c759b712-4932-513a-b6a1-f86c0bcc1a5e"
}
```

1.7.2 Product record

A product record represents a CPE-relevant product or platform.

Fields:

- `uuid`: stable UUID for the product.
- `vendor_uuid`: stable UUID of the associated vendor.
- `name`: normalized machine-oriented product name.
- `title`: human-readable product title; may be **`null`**.
- `notes`: optional registry notes; may be **`null`**.
- `created_at`: creation timestamp.
- `updated_at`: last update timestamp.

Example:

```
{
  "created_at": "2026-04-26T14:26:10.000000",
  "name": "misp",
  "notes": null,
  "title": "Misp",
  "updated_at": "2026-04-26T14:26:10.000000",
  "uuid": "f28e0890-4e1b-59a5-b52c-3e9354071db7",
  "vendor_uuid": "536016cd-a314-5880-947a-e465002d0fab"
}
```

1.7.3 CPE entry record

A CPE entry record stores the CPE-compatible representation and parsed CPE attributes.

Fields:

- `cpe_uri`: full CPE 2.3 URI, such as `cpe:2.3:a:misp:misp:*:*:*:*:*:*:*:*`.
- `cpe_name_id`: upstream CPE name UUID when available, such as an NVD `cpeNameId`; may be **`null`**.
- `vendor_uuid`: UUID of the associated vendor record.
- `product_uuid`: UUID of the associated product record.
- `deprecated`: boolean deprecation flag.
- `deprecated_by`: replacement CPE URI or reference when available; may be **`null`**.
- `part`: CPE part, commonly `a`, `o`, or `h`.
- `version`: CPE version component, or `*`.
- `update`: CPE update component, or `*`.
- `edition`: CPE edition component, or `*`.
- `language`: CPE language component, or `*`.
- `sw_edition`: CPE software edition component, or `*`.
- `target_sw`: CPE target software component, or `*`.
- `target_hw`: CPE target hardware component, or `*`.
- `other`: CPE other component, or `*`.

- `title`: human-readable title; may be **`null`**.
- `notes`: optional notes; may be **`null`**.
- `from_proposal`: boolean indicating whether the entry originated from a proposal workflow.
- `created_at`: creation timestamp.
- `updated_at`: last update timestamp.

Example:


```
{
  "cpe_name_id": null,
  "cpe_uri": "cpe:2.3:a:misp:misp:*:*:*:*:*:*:*:*",
  "created_at": "2026-04-26T14:26:30.000000",
  "deprecated": false,
  "deprecated_by": null,
  "edition": "*",
  "from_proposal": false,
  "language": "*",
  "notes": "Imported from NVD CPE 2.0 feed",
  "other": "*",
  "part": "a",
  "product_uuid": "f28e0890-4e1b-59a5-b52c-3e9354071db7",
  "sw_edition": "*",
  "target_hw": "*",
  "target_sw": "*",
  "title": "Misp",
  "update": "*",
  "updated_at": "2026-04-26T14:26:30.000000",
  "vendor_uuid": "536016cd-a314-5880-947a-e465002d0fab",
  "version": "*"
}
```


1.7.4 Metadata record

A metadata record attaches a namespaced key/value pair to a vendor or product record.

Fields:

- `metadata_key`: namespaced metadata key. GCVE-defined keys MUST use the `gcve:` prefix.
- `metadata_value`: metadata value as a string.
- `record_type`: target record type, such as `vendor` or `product`.
- `record_uuid`: UUID of the target vendor or product record.
- `submitter_name`: submitter name; may be an empty string or **`null`**.
- `submitter_email`: submitter email; may be an empty string or **`null`**.
- `submitted_at`: submission timestamp.
- `approved_at`: approval timestamp; may be **`null`** if not approved.

- `created_at`: creation timestamp.
- `updated_at`: last update timestamp.

Example metadata record for a vendor URL:


```
{
  "approved_at": "2026-04-26T16:37:57.912577",
  "created_at": "2026-04-26T16:37:57.919754",
  "metadata_key": "gcve:url",
  "metadata_value": "https://misp-project.org/",
  "record_type": "vendor",
  "record_uuid": "536016cd-a314-5880-947a-e465002d0fab",
  "submitted_at": "2026-04-26T16:36:55.583569",
  "submitter_email": "",
  "submitter_name": "",
  "updated_at": "2026-04-26T16:37:57.919758"
}
```


Example metadata record for a vendor description:


```
{
  "approved_at": "2026-04-26T16:38:01.609044",
  "created_at": "2026-04-26T16:38:01.612249",
  "metadata_key": "gcve:description",
  "metadata_value": "MISP - Threat Intelligence Sharing Platform\r\n\r\
     nMISP is an open source software solution for collecting, storing,
      distributing and sharing cyber security indicators and threats
     about cyber security incidents analysis and malware analysis. MISP
      is designed by and for incident analysts, security and ICT
     professionals or malware reversers to support their day-to-day
     operations to share structured information efficiently.\r\n\r\nThe
      objective of MISP is to foster the sharing of structured
     information within the security community and abroad. MISP
     provides functionalities to support the exchange of information
     but also the consumption of said information by Network Intrusion
     Detection Systems (NIDS), LIDS but also log analysis tools, SIEMs.
     ",
  "record_type": "vendor",
  "record_uuid": "536016cd-a314-5880-947a-e465002d0fab",
  "submitted_at": "2026-04-26T16:37:49.951062",
  "submitter_email": "",
  "submitter_name": "",
  "updated_at": "2026-04-26T16:38:01.612251"
}
```


1.7.5 Recommended GCVE metadata keys

The following GCVE metadata keys are recommended for CPE-related use:

- `gcve:url`: primary public URL for a vendor or product.
- `gcve:description`: human-readable description.
- `gcve:source`: source authority or import origin for the assertion.
- `gcve:canonical`: string value indicating whether the associated assertion is canonical, such as **`true`** or **`false`**.
- `gcve:deprecated`: string value indicating whether a record or assertion is deprecated, such as **`true`** or **`false`**.
- `gcve:replaced-by`: UUID of the preferred replacement record, when applicable.
- `gcve:external-id`: external identifier associated with a vendor or product.
- `gcve:homepage`: synonym for or more specific variant of `gcve:url`, when a registry policy distinguishes homepage from other URLs.

CPE 2.3 strings SHOULD be represented as CPE entry records in the `cpes` array. Metadata MAY still be used for auxiliary CPE-related assertions, but implementations SHOULD NOT rely on metadata as the primary storage for imported CPE entries when the `cpes` array is available.

Implementations MAY define additional metadata keys under other namespace prefixes. They MUST NOT redefine the semantics of GCVE-defined `gcve:` keys.

1.7.6 Relationship record

A relationship record links one vendor or product record to another vendor or product record.

Fields:

- `relationship_type`: relationship type.
- `source_vendor_uuid`: source vendor UUID, or **`null`**.
- `source_product_uuid`: source product UUID, or **`null`**.
- `target_vendor_uuid`: target vendor UUID, or **`null`**.
- `target_product_uuid`: target product UUID, or **`null`**.
- `rationale`: free-text rationale for the relationship; may be an empty string or **`null`**.
- `submitter_name`: submitter name; may be an empty string or **`null`**.
- `submitter_email`: submitter email; may be an empty string or **`null`**.
- `submitted_at`: submission timestamp; may be **`null`**.
- `approved_at`: approval timestamp; may be **`null`**.
- `created_at`: creation timestamp.
- `updated_at`: last update timestamp.

Rules:

- exactly one of `source_vendor_uuid` or `source_product_uuid` SHOULD be non-null;

- exactly one of `target_vendor_uuid` or `target_product_uuid` SHOULD be non-null;
- vendor-to-vendor relationships SHOULD use `source_vendor_uuid` and `target_vendor_uuid`;
- product-to-product relationships SHOULD use `source_product_uuid` and `target_product_uuid`;
- cross-type relationships MAY be used only when explicitly supported by registry policy;
- relationship records do not require a standalone `relationshipId` in the `cpe-editor` exchange format;
- importers SHOULD treat (`source_vendor_uuid`, `source_product_uuid`, `target_vendor_uuid`, `target_product_uuid`, `relationship_type`) as the natural relationship identity.

Example vendor synonym relationship:


```
{
  "approved_at": "2026-04-26T15:00:15.404458",
  "created_at": "2026-04-26T15:00:24.654104",
  "rationale": "",
  "relationship_type": "synonym-of",
  "source_product_uuid": null,
  "source_vendor_uuid": "536016cd-a314-5880-947a-e465002d0fab",
  "submitted_at": "2026-04-26T14:59:36.958957",
  "submitter_email": "",
  "submitter_name": "",
  "target_product_uuid": null,
  "target_vendor_uuid": "c759b712-4932-513a-b6a1-f86c0bcc1a5e",
  "updated_at": "2026-04-26T15:00:24.654105"
}
```


Example product rename relationship:


```
{
  "approved_at": "2026-04-26T15:28:49.747406",
  "created_at": "2026-04-26T15:28:51.283869",
  "rationale": "",
  "relationship_type": "renamed-to",
  "source_product_uuid": "7669a502-7617-50ad-a2d3-1b5f36b46051",
  "source_vendor_uuid": null,
  "submitted_at": "2026-04-26T15:28:37.806590",
  "submitter_email": "",
  "submitter_name": "",
  "target_product_uuid": "f28e0890-4e1b-59a5-b52c-3e9354071db7",
  "target_vendor_uuid": null,
  "updated_at": "2026-04-26T15:28:51.283871"
}
```

1.8 Recommended relationship types

The following relationship types are defined:

- `synonym-of`: the source record is a synonym of the target record.
- `canonical-of`: the source record is an alias whose canonical record is the target.
- `renamed-to`: the source record has been renamed to the target.
- `superseded-by`: the source record is obsolete and replaced operationally by the target.
- `equivalent-to`: both records are considered operationally equivalent for identification purposes.
- `vendor-merge-into`: the source vendor has merged into the target vendor.
- `derived-from`: the source record was derived from another source record.

Implementations may define additional relationship types, but they should not change the semantics of the above values.

1.9 Compatibility model

GCVE-BCP-10 does not replace the CPE string format. Instead, it layers stable registry identity, namespaced metadata, CPE entry records, and relationship semantics on top of it.

A GCVE-BCP-10 implementation:

- **must** retain CPE-compatible name or match strings where CPE matching is required;
- **must not** require existing CPE parsers to understand GCVE UUIDs, metadata records, or relationship records to parse the CPE string itself;
- **may** provide registry metadata beyond what legacy CPE consumers understand;
- **should** allow lossless round-trip preservation of the original CPE-compatible string;
- **should** store imported CPE names as `cpes` records linked to vendor and product UUIDs;
- **may** use metadata for descriptive or auxiliary assertions that are not core CPE entries.

This means existing tools can continue using the CPE string for matching, while GCVE-aware systems use UUIDs, metadata, CPE records, and relationships for lifecycle-safe management.

1.10 Proposed schema

The following JSON Schema describes the GCVE-BCP-10 registry exchange format aligned with the `cpe-editor` dataset format.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "JSON Schema for GCVE-BCP-10 cpe-editor Dataset Exchange",
  "$id": "https://gcve.eu/schema/gcve-bcp-10/cpe_editor_dataset_json.
     schema",
  "definitions": {
    "uuid": {
      "type": "string",
      "format": "uuid"
    },
    "timestamp": {
      "type": "string",
      "format": "date-time"
    },
    "nullable_timestamp": {
      "anyOf": [
        { "$ref": "#/definitions/timestamp" },
        { "type": "null" }
      ]
    },
    "nullable_string": {
      "type": ["string", "null"]
    },
    "def_counts": {
      "type": "object",
      "properties": {
        "vendors": { "type": "integer" },
        "products": { "type": "integer" },
        "cpes": { "type": "integer" },
        "metadata": { "type": "integer" },
        "relationships": { "type": "integer" },
        "proposals": { "type": "integer" }
      },
      "required": [
        "vendors",
        "products",
        "cpes",
        "metadata",
        "relationships",
        "proposals"
      ],
      "additionalProperties": false
    },
    "def_vendor": {
      "type": "object",
      "properties": {
        "uuid": { "$ref": "#/definitions/uuid" },
        "name": { "type": "string" },
        "title": { "$ref": "#/definitions/nullable_string" },
        "notes": { "$ref": "#/definitions/nullable_string" },
```

```
        "created_at": { "$ref": "#/definitions/timestamp" },
        "updated_at": { "$ref": "#/definitions/timestamp" }
      },
      "required": [
        "created_at",
        "name",
        "notes",
        "title",
        "updated_at",
        "uuid"
      ],
      "additionalProperties": false
    },
    "def_product": {
      "type": "object",
      "properties": {
        "uuid": { "$ref": "#/definitions/uuid" },
        "vendor_uuid": { "$ref": "#/definitions/uuid" },
        "name": { "type": "string" },
        "title": { "$ref": "#/definitions/nullable_string" },
        "notes": { "$ref": "#/definitions/nullable_string" },
        "created_at": { "$ref": "#/definitions/timestamp" },
        "updated_at": { "$ref": "#/definitions/timestamp" }
      },
      "required": [
        "created_at",
        "name",
        "notes",
        "title",
        "updated_at",
        "uuid",
        "vendor_uuid"
      ],
      "additionalProperties": false
    },
    "def_cpe": {
      "type": "object",
      "properties": {
        "cpe_uri": { "type": "string" },
        "cpe_name_id": {
          "anyOf": [
            { "$ref": "#/definitions/uuid" },
            { "type": "null" }
          ]
        },
        "vendor_uuid": { "$ref": "#/definitions/uuid" },
        "product_uuid": { "$ref": "#/definitions/uuid" },
        "deprecated": { "type": "boolean" },
        "deprecated_by": { "$ref": "#/definitions/nullable_string" },
        "part": { "type": "string", "enum": ["a", "o", "h", "*"] },
        "version": { "type": "string" },
```

```
        "update": { "type": "string" },
        "edition": { "type": "string" },
        "language": { "type": "string" },
        "sw_edition": { "type": "string" },
        "target_sw": { "type": "string" },
        "target_hw": { "type": "string" },
        "other": { "type": "string" },
        "title": { "$ref": "#/definitions/nullable_string" },
        "notes": { "$ref": "#/definitions/nullable_string" },
        "from_proposal": { "type": "boolean" },
        "created_at": { "$ref": "#/definitions/timestamp" },
        "updated_at": { "$ref": "#/definitions/timestamp" }
      },
      "required": [
        "cpe_name_id",
        "cpe_uri",
        "created_at",
        "deprecated",
        "deprecated_by",
        "edition",
        "from_proposal",
        "language",
        "notes",
        "other",
        "part",
        "product_uuid",
        "sw_edition",
        "target_hw",
        "target_sw",
        "title",
        "update",
        "updated_at",
        "vendor_uuid",
        "version"
      ],
      "additionalProperties": false
    },
    "def_metadata": {
      "type": "object",
      "properties": {
        "metadata_key": {
          "type": "string",
          "pattern": "^[a-zA-Z][a-zA-Z0-9_-]*:[A-Za-z0-9._-]+$"
        },
        "metadata_value": { "type": "string" },
        "record_type": { "type": "string", "enum": ["vendor", "product"
            ] },
        "record_uuid": { "$ref": "#/definitions/uuid" },
        "submitter_name": { "$ref": "#/definitions/nullable_string" },
        "submitter_email": { "$ref": "#/definitions/nullable_string" },
        "submitted_at": { "$ref": "#/definitions/nullable_timestamp" },
```

```
        "approved_at": { "$ref": "#/definitions/nullable_timestamp" },
        "created_at": { "$ref": "#/definitions/nullable_timestamp" },
        "updated_at": { "$ref": "#/definitions/nullable_timestamp" }
      },
      "required": [
        "approved_at",
        "created_at",
        "metadata_key",
        "metadata_value",
        "record_type",
        "record_uuid",
        "submitted_at",
        "submitter_email",
        "submitter_name",
        "updated_at"
      ],
      "additionalProperties": false
    },
    "def_relationship": {
      "type": "object",
      "properties": {
        "relationship_type": {
          "type": "string",
          "enum": [
            "synonym-of",
            "canonical-of",
            "renamed-to",
            "superseded-by",
            "equivalent-to",
            "vendor-merge-into",
            "derived-from"
          ]
        },
        "source_vendor_uuid": {
          "anyOf": [
            { "$ref": "#/definitions/uuid" },
            { "type": "null" }
          ]
        },
        "source_product_uuid": {
          "anyOf": [
            { "$ref": "#/definitions/uuid" },
            { "type": "null" }
          ]
        },
        "target_vendor_uuid": {
          "anyOf": [
            { "$ref": "#/definitions/uuid" },
            { "type": "null" }
          ]
        },
```

```
        "target_product_uuid": {
          "anyOf": [
            { "$ref": "#/definitions/uuid" },
            { "type": "null" }
          ]
        },
        "rationale": { "$ref": "#/definitions/nullable_string" },
        "submitter_name": { "$ref": "#/definitions/nullable_string" },
        "submitter_email": { "$ref": "#/definitions/nullable_string" },
        "submitted_at": { "$ref": "#/definitions/nullable_timestamp" },
        "approved_at": { "$ref": "#/definitions/nullable_timestamp" },
        "created_at": { "$ref": "#/definitions/nullable_timestamp" },
        "updated_at": { "$ref": "#/definitions/nullable_timestamp" }
      },
      "required": [
        "approved_at",
        "created_at",
        "rationale",
        "relationship_type",
        "source_product_uuid",
        "source_vendor_uuid",
        "submitted_at",
        "submitter_email",
        "submitter_name",
        "target_product_uuid",
        "target_vendor_uuid",
        "updated_at"
      ],
      "additionalProperties": false
    }
  },
  "type": "object",
  "properties": {
    "format": { "type": "string", "const": "cpe-editor-dataset" },
    "version": { "type": "string" },
    "exported_at": { "$ref": "#/definitions/timestamp" },
    "counts": { "$ref": "#/definitions/def_counts" },
    "vendors": {
      "type": "array",
      "items": { "$ref": "#/definitions/def_vendor" }
    },
    "products": {
      "type": "array",
      "items": { "$ref": "#/definitions/def_product" }
    },
    "cpes": {
      "type": "array",
      "items": { "$ref": "#/definitions/def_cpe" }
    },
    "metadata": {
      "type": "array",
```

```
253         "items": { "$ref": "#/definitions/def_metadata" }
254       },
255       "relationships": {
256         "type": "array",
257         "items": { "$ref": "#/definitions/def_relationship" }
258       },
259       "proposals": {
260         "description": "Implementation-specific moderation history. It
              may be empty and is not required for minimal BCP
              interoperability.",
261         "type": "array",
262         "items": { "type": "object" }
263       }
264     },
265     "required": [
266       "format",
267       "version",
268       "exported_at",
269       "counts",
270       "vendors",
271       "products",
272       "cpes",
273       "metadata",
274       "relationships",
275       "proposals"
276     ],
277     "additionalProperties": false
278   }
```

1.11 Field semantics

1.11.1 uuid

A stable UUID identifying a vendor or product record independently of textual names, titles, aliases, and CPE-compatible strings.

Rules:

- `uuid` MUST remain stable across repeated imports of the same normalized vendor or product identity.
- Vendor and product UUIDs SHOULD be deterministic UUIDv5 values generated as specified in the UUID generation section.
- A new `uuid` MUST NOT be assigned solely because the display title, notes, metadata, or CPE-compatible string changes.

- If two records are merged and declared to refer to the same identity, registry policy determines which UUID remains canonical and which becomes deprecated or related through a lifecycle relationship.

1.11.2 `name`

The normalized machine-oriented name for a vendor or product.

Rules:

- `name` SHOULD be lowercase where practical.
- `name` SHOULD use stable separators such as underscores or hyphens according to registry policy.
- `name` SHOULD NOT be used as the durable identifier by consumers.
- Changes to `name` MUST NOT break UUID-based references.

1.11.3 `title`

The human-readable display title for a vendor or product.

Rules:

- `title` MAY preserve capitalization and spacing preferred by the vendor or registry.
- `title` MAY change over time.
- `title` MUST NOT be used as the durable identifier.

1.11.4 `vendor_uuid`

The UUID of the vendor associated with a product or CPE entry.

Rules:

- In a product record, `vendor_uuid` MUST reference an existing vendor record.
- In a CPE entry record, `vendor_uuid` MUST reference the vendor that corresponds to the vendor component of the CPE string.

1.11.5 `product_uuid`

The UUID of the product associated with a CPE entry.

Rules:

- `product_uuid` MUST reference an existing product record.
- A CPE entry MUST link to both a vendor and a product.

1.11.6 `cpe_uri`

The full CPE-compatible URI.

Rules:

- `cpe_uri` SHOULD be a CPE 2.3 formatted string.
- `cpe_uri` SHOULD be preserved losslessly from the source dataset where possible.
- Importers SHOULD match existing CPE entries by `cpe_uri` first.

1.11.7 `cpe_name_id`

An optional upstream CPE name identifier.

Rules:

- `cpe_name_id` MAY be populated from upstream sources such as NVD `cpeNameId`.
- If `cpe_uri` matching fails, importers MAY use `cpe_name_id` as a secondary matching key.
- `cpe_name_id` is not a replacement for vendor and product UUIDs.

1.11.8 `deprecated` and `deprecated_by`

Lifecycle controls used when a CPE entry is retired.

Rules:

- Deprecated CPE entries SHOULD remain resolvable.
- `deprecated_by` SHOULD reference a preferred replacement CPE URI or upstream replacement reference when available.

1.11.9 `metadata_key`

A namespaced key identifying the type of metadata assertion.

Rules:

- `metadata_key` MUST include a namespace prefix followed by a colon.
- GCVE-defined metadata keys MUST use the `gcve:` prefix.

- Non-GCVE metadata keys MAY use other namespace prefixes.
- Implementations MUST NOT treat unrecognized metadata keys as matching criteria unless explicitly configured to do so.

1.11.10 `metadata_value`

The value associated with `metadata_key`.

Rules:

- `metadata_value` is represented as a string.
- Implementations MAY interpret selected GCVE metadata values according to registry policy.
- Descriptive information such as URLs and descriptions SHOULD be represented as metadata.

1.11.11 `record_type` and `record_uuid`

These fields identify the target of a metadata assertion.

Rules:

- `record_type` SHOULD be `vendor` or `product`.
- `record_uuid` MUST reference the UUID of the corresponding record type.
- Metadata for a vendor MUST use `record_type: "vendor"`.
- Metadata for a product MUST use `record_type: "product"`.

1.11.12 `relationship_type`

The typed semantics of a relationship between two records.

Rules:

- `relationship_type` MUST describe how the source record relates to the target record.
- Implementations MUST preserve the direction of relationships.
- `renamed-to` and `superseded-by` are directional.
- `equivalent-to` may be treated as symmetric by consumers, but the stored relationship still has a source and target.

1.11.13 `source_vendor_uuid`, `source_product_uuid`, `target_vendor_uuid`, and `target_product_uuid`

These fields identify the source and target records for a relationship.

Rules:

- source fields identify the record from which the relationship starts;
- target fields identify the record to which the relationship points;
- unused source and target fields MUST be **`null`**;
- vendor-to-vendor relationships use vendor UUID fields;
- product-to-product relationships use product UUID fields.

1.11.14 `submitted_at`, `approved_at`, `created_at`, and `updated_at`

Audit timestamps for registry operations.

Rules:

- `submitted_at` records when an assertion was submitted.
- `approved_at` records when an assertion was approved for registry use.
- `created_at` records when the registry object was created.
- `updated_at` records the latest update to the registry object.
- Timestamps SHOULD be represented as ISO 8601 date-time strings.
- `approved_at` and `submitted_at` MAY be **`null`** for imported or unapproved implementation records.

1.12 Registry behavior

GCVE operates a registry to support updates, handle synonyms, manage vendors and products, synchronize CPE entries, and synchronize metadata and relationships.

1.12.1 Registry responsibilities

The registry:

- assigns and preserves vendor and product UUIDs;
- generates deterministic UUIDv5 identifiers for imported vendor and product records;
- tracks canonical names and aliases;
- manages synonym and rename relationships;
- supports vendor normalization and vendor lifecycle changes;
- stores CPE-compatible names as CPE entry records;
- records submitter, approval, creation, and update timestamps where applicable;
- exposes updates suitable for synchronization by external systems.

1.12.2 Vendor management

GCVE may maintain a distinct vendor registry, including stable vendor UUIDs.

Vendor management may include:

- normalization of spelling variants;
- company mergers and acquisitions;
- rebranding;
- separation of legal vendor identity from naming conventions used in CPE-compatible strings;
- descriptive metadata such as `gcve:url` and `gcve:description`.

1.12.3 Product management

GCVE may maintain a distinct product registry, including stable product UUIDs.

Product management may include:

- product renaming;
- historical names;
- community versus vendor-preferred naming;
- imported external identifiers;
- spelling normalization;
- product line consolidation;
- CPE-compatible names attached through CPE entry records.

1.12.4 CPE entry management

CPE entry management may include:

- importing CPE entries from the NVD CPE 2.0 feed;
- importing concrete CPE names from the NVD CPE Match feed;
- parsing CPE 2.3 strings into component fields;
- preserving `cpeNameId` when present;
- preserving deprecation and replacement information;
- linking each CPE entry to deterministic vendor and product UUIDs.

1.12.5 Synonym handling

GCVE should support synonym management for cases such as:

- vendor aliases;
- product aliases;
- historical names;
- imported external identifiers;
- spelling normalization;
- product line consolidation.

Synonyms should not require changing the stable UUID when the underlying identity is preserved.

1.12.6 Resolution expectations

Given any known UUID or recognized CPE-compatible alias, the registry should be able to return:

- the current canonical vendor or product record;
- the stable UUID;
- known synonyms;
- lifecycle status;
- related metadata records;
- linked CPE entries where available.

1.13 Matching behavior

GCVE-BCP-10 does not change the core CPE applicability concept. Matching is still performed using CPE-compatible criteria and version range semantics.

However, GCVE-aware systems may enhance matching by:

- resolving vendor and product synonyms before comparison;
- canonicalizing renamed products;
- dereferencing deprecated records to active replacements;
- preserving provenance and approval state during matching output;
- mapping matched CPE-compatible strings back to stable product and vendor UUIDs.

This improves consistency across datasets that use different but equivalent product or vendor names.

1.14 Example registry bundle

The following example shows a `cpe-editor`-aligned dataset bundle using vendor records, product records, CPE entries, metadata records, and relationships.


```
{
  "counts": {
    "cpes": 1,
    "metadata": 2,
    "products": 1,
    "proposals": 0,
    "relationships": 1,
    "vendors": 2
  },
  "cpes": [
    {
      "cpe_name_id": null,
      "cpe_uri": "cpe:2.3:a:misp:misp:*:*:*:*:*:*:*:*",
      "created_at": "2026-04-26T14:26:30.000000",
      "deprecated": false,
      "deprecated_by": null,
      "edition": "*",
      "from_proposal": false,
      "language": "*",
      "notes": "Imported from NVD CPE 2.0 feed",
      "other": "*",
      "part": "a",
      "product_uuid": "f28e0890-4e1b-59a5-b52c-3e9354071db7",
      "sw_edition": "*",
      "target_hw": "*",
      "target_sw": "*",
      "title": "Misp",
      "update": "*",
      "updated_at": "2026-04-26T14:26:30.000000",
      "vendor_uuid": "536016cd-a314-5880-947a-e465002d0fab",
      "version": "*"
    }
  ],
  "exported_at": "2026-04-26T17:00:00+00:00",
  "format": "cpe-editor-dataset",
  "metadata": [
    {
      "approved_at": "2026-04-26T16:37:57.912577",
      "created_at": "2026-04-26T16:37:57.919754",
      "metadata_key": "gcve:url",
      "metadata_value": "https://misp-project.org/",
      "record_type": "vendor",
      "record_uuid": "536016cd-a314-5880-947a-e465002d0fab",
      "submitted_at": "2026-04-26T16:36:55.583569",
      "submitter_email": "",
```

```
      "submitter_name": "",
      "updated_at": "2026-04-26T16:37:57.919758"
    },
    {
      "approved_at": "2026-04-26T16:38:01.609044",
      "created_at": "2026-04-26T16:38:01.612249",
      "metadata_key": "gcve:description",
      "metadata_value": "MISP - Threat Intelligence Sharing Platform\r\n\r\nMISP is an open source software solution for collecting, storing, distributing and sharing cyber security indicators and threats about cyber security incidents analysis and malware analysis. MISP is designed by and for incident analysts, security and ICT professionals or malware reversers to support their day-to-day operations to share structured information efficiently.\r\n\r\nThe objective of MISP is to foster the sharing of structured information within the security community and abroad. MISP provides functionalities to support the exchange of information but also the consumption of said information by Network Intrusion Detection Systems (NIDS), LIDS but also log analysis tools, SIEMs.",
      "record_type": "vendor",
      "record_uuid": "536016cd-a314-5880-947a-e465002d0fab",
      "submitted_at": "2026-04-26T16:37:49.951062",
      "submitter_email": "",
      "submitter_name": "",
      "updated_at": "2026-04-26T16:38:01.612251"
    }
  ],
  "products": [
    {
      "created_at": "2026-04-26T14:26:10.000000",
      "name": "misp",
      "notes": null,
      "title": "Misp",
      "updated_at": "2026-04-26T14:26:10.000000",
      "uuid": "f28e0890-4e1b-59a5-b52c-3e9354071db7",
      "vendor_uuid": "536016cd-a314-5880-947a-e465002d0fab"
    }
  ],
  "proposals": [],
  "relationships": [
    {
      "approved_at": "2026-04-26T15:00:15.404458",
      "created_at": "2026-04-26T15:00:24.654104",
      "rationale": "",
      "relationship_type": "synonym-of",
      "source_product_uuid": null,
      "source_vendor_uuid": "536016cd-a314-5880-947a-e465002d0fab",
      "submitted_at": "2026-04-26T14:59:36.958957",
      "submitter_email": "",
      "submitter_name": "",
```

```
      "target_product_uuid": null,
      "target_vendor_uuid": "c759b712-4932-513a-b6a1-f86c0bcc1a5e",
      "updated_at": "2026-04-26T15:00:24.654105"
    }
  ],
  "vendors": [
    {
      "created_at": "2026-04-26T14:25:34.846011",
      "name": "misp",
      "notes": null,
      "title": "Misp",
      "updated_at": "2026-04-26T14:25:34.846011",
      "uuid": "536016cd-a314-5880-947a-e465002d0fab"
    },
    {
      "created_at": "2026-04-26T14:24:53.155776",
      "name": "misp-project",
      "notes": null,
      "title": "Misp Project",
      "updated_at": "2026-04-26T14:24:53.155776",
      "uuid": "c759b712-4932-513a-b6a1-f86c0bcc1a5e"
    }
  ],
  "version": "1"
}
```


1.15 Migration considerations

To migrate an existing CPE-based dataset into GCVE-BCP-10:

- preserve existing CPE-compatible strings in `cpes.cpe_uri`;
- parse CPE 2.3 fields into the corresponding `cpes` component fields;
- assign deterministic UUIDv5 values to vendor records using the UUID generation profile above;
- assign deterministic vendor-scoped UUIDv5 values to product records using the UUID generation profile above;
- preserve upstream `cpeNameId` as `cpe_name_id` when available;
- preserve upstream deprecation information as `deprecated` and `deprecated_by` when available;
- attach descriptive source information using metadata such as `gcve:source` when needed;
- detect aliases and represent them as relationship records;
- identify canonical names where multiple equivalent names exist;
- keep deprecated CPE entries resolvable through `deprecated_by`, metadata, or relationship chains.

Migration should prefer stable identity over textual immutability.

1.16 Interoperability guidance

Implementations that do not understand GCVE-BCP-10 registry records may ignore fields and records such as:

- vendor and product UUIDs;
- metadata records;
- relationship records;
- approval and submission timestamps;
- submitter fields;
- lifecycle metadata;
- optional proposal history.

As long as CPE-compatible strings stored in `cpes.cpe_uri` remain valid, interoperability with legacy parsers is preserved at the string level.

1.17 Security and integrity considerations

Consumers should not automatically trust synonym, rename, vendor merge, product merge, CPE, or metadata assertions from untrusted sources.

Implementations should consider:

- provenance validation;
- registry authenticity;
- update signing;
- auditability of metadata and relationship changes;
- conflict handling when multiple authorities disagree about canonicalization;
- validation of namespaced metadata keys;
- validation that relationship endpoints reference existing records;
- validation that CPE entry vendor/product links match parsed CPE vendor/product components;
- validation that deterministic UUIDs match the documented UUIDv5 profile before accepting imported records as canonical.

Stable UUIDs reduce ambiguity, but trust still depends on the authority maintaining the registry.

1.17.1 Vulnerability reference

A vulnerability reference links a CPE entry (or product context) to a vulnerability identifier such as GCVE.

Fields:

- `vulnerability_id`: identifier of the vulnerability (e.g. `gcve-1-2026-0024`).
- `vulnerability_source`: source namespace (e.g. `GCVE`).
- `cpe_applicability`: applicability status such as `vulnerable`, `not_affected`, or `unknown`.
- `submitted_at`: submission timestamp.
- `approved_at`: approval timestamp.
- `rationale`: optional free-text explanation.
- `id`: local identifier.

This allows direct linkage between platform enumeration and vulnerability intelligence, enabling:

- precise applicability mapping;
- integration with vulnerability lookup systems;
- decentralized enrichment by GNAs.

Example extension in a CPE record:

```
{
  "cpe_uri": "cpe:2.3:a:misp:misp:2.1.18:*:*:*:*:*:*:*",
  "product_uuid": "f28e0890-4e1b-59a5-b52c-3e9354071db7",
  "vendor_uuid": "536016cd-a314-5880-947a-e465002d0fab",
  "version": "2.1.18",
  "vulnerability_references": [
    {
      "id": 1,
      "vulnerability_id": "gcve-1-2026-0024",
      "vulnerability_source": "GCVE",
      "cpe_applicability": "vulnerable",
      "submitted_at": "2026-04-28T13:55:45.452205",
      "approved_at": "2026-04-28T13:55:57.956959",
      "rationale": ""
    }
  ]
}
```

1.18 Summary

GCVE-BCP-10 improves CPE operations without discarding CPE compatibility. It introduces:

- deterministic UUIDv5 vendor and product UUIDs;
- a documented namespace hierarchy rooted in `uuid5(NAMESPACE_URL, "GCVE-BCP-10")`;
- first-class `vendors`, `products`, and `cpes` records aligned with `cpe-editor`;
- namespaced metadata records using keys such as `gcve:url` and `gcve:description`;
- structured synonym, rename, equivalence, and lifecycle relationships;
- registry-based vendor, product, CPE, and lifecycle management;
- audit-oriented submission, approval, creation, and update timestamps.

These extensions make platform identification more durable, auditable, and maintainable in real-world vulnerability and asset management workflows.

1.19 Suggested minimal compliance profile

An implementation conforms to the GCVE-BCP-10 minimal profile if it:

- supports CPE-compatible strings;
- exports `format: "cpe-editor-dataset"` or can transform losslessly to that shape;
- assigns deterministic UUIDv5 values to vendor and product records according to this document;
- stores imported CPE names as `cpes` records linked to vendor and product UUIDs;
- supports metadata records with namespaced metadata keys;
- uses the `gcve:` prefix for GCVE-defined metadata keys;
- can express at least `synonym-of` and `renamed-to` relationships;
- preserves deprecated records without breaking resolution.

A full-featured GCVE registry implementation should additionally support:

- vendor normalization;
- product normalization;
- canonical name resolution;
- replacement and deprecation workflows;
- history tracking for metadata and relationship changes;
- synchronization of vendors, products, CPE entries, metadata, and relationships;
- validation of deterministic UUIDv5 values on import.

2 Acknowledgements

2.1 BCP-10 Coordinator

- Alexandre Dulaunoy, CIRCL

2.2 Contributions

The GCVE initiative gratefully acknowledges the substantial contributions from the following individuals via public review:

- Jerry Gamblin
- Quentin Jerome
- Contributors during the hackathon.lu