KOALA: Transparency Exchange API Overview

oej wg 2024-09-26 v1.2

Olle E Johansson - oej@edvina.net

OWASP®

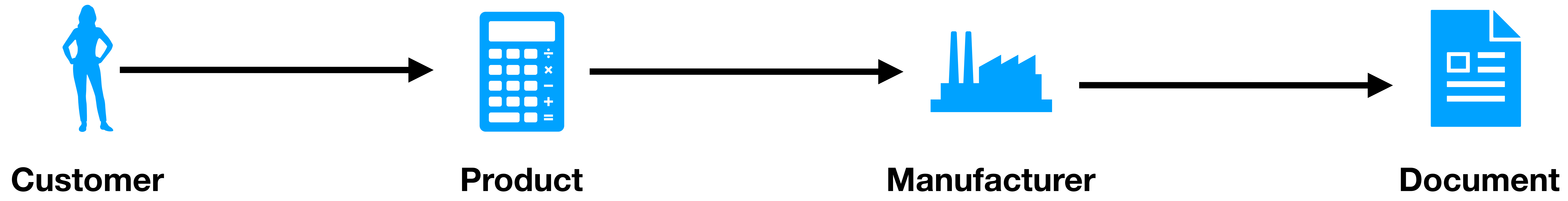
CycloneDX

Project Koala

The problem

Many customers have many products from many vendors.
In order to automatically or manually be able to retrieve standardised transparency attestations (SBOM, VEX and others) we need to also standardise discovery, identification, authentication and retrieval of these documents.

The solution has to scale globally.



TEA will be a standardised API

- The Transparency Exchange API will standardise publication and retrieval of transparency data for software and hardware.
- The API is based on a discovery scheme with a URN called Transparency Exchange Identifier
- The URN uniqueness is based on DNS names and existing product identifiers, from EAN barcodes to Package URLs and random UUIDs. The manufacturer defines the TEI.
- While TEA is standardised as part of the OWASP CycloneDX project, it's not specific to the CycloneDX format.
- TEA includes authentication and authorization, controlling who can access what information and who can publish what.

TEA Components

Product

A product is something sold with software - an app, a server, embedded system, toy, IoT sensor etc

TEX Product index

TPI is a list of parts in a product, called TEA leaves. For each part there is a pointer (TEI or URL) for the TLI for each leaf.

TEX Leaf index

TLI is an index of all software versions for a product with indications of the state of the software version and reference to where a collection can be found.

TEI

TEI is a Transparency Exchange Identifier. A unique identifier for a specific product regardless of software version.
TEI is based on existing identities for a product, not replacing them.

TEA Collection

The TEA collection is the repository of current artifacts for a specific version of software in a given product.
The collection includes SBOM, VEX, SLSA attestations and more.

TEA Use cases

https://github.com/CycloneDX/transparency-exchange-api/blob/main/doc/tea-usecases.md

Use case #1: Retail consumer

- Alice bought a gadget at the gadget store that contains a full Linux system. Where will she find the SBOM and VEX for the gadget?

Use case #2: Business consumer

- Acme LLC buys 3 000 gadgetrons from Emca LTD to be distributed over a retail chain. Acme runs an in-house vulnerability management system (Dependency Track) to manage SBOMs and download VEX files to check for vulnerabilities. Acme has products from exactly 14.385 vendors in the system.
- How will their systems get continuous access to current and old documents - attestations, SBOM, VEX and other files?

Use case #3: Regulators

- Alice & Bob Enterprises AB has gotten a EUCC certification to get their Whola Firewall certified for CRA-compatible CE labeling. In order to maintain the certification the certifying body needs access to SBOM and VEX updates from A&BE in an automated way.

Use case #4: Potential customer

- Palme Auditors INC wants to buy the ACME SWISH product from a vendor. They want to examine vulnerability handling and get some insights into the products before making a decision.

Use case #5: Open Source user

- Palme Inc considers using the asterisk.org open source telephony PBX.
- They need to make an assessment before starting tests and possible production use.
- The can either use the Debian package, the Alpine Linux Package or build a binary themselves from source code.
- How can they find the transparency exchange data sets?

#6: Components for including in products by developers

- A **developer** needs a component - software, library (open source and/or proprietary) to include in a product - publicly available (may be commercial) or in an internal system
- Developer can be individual, company or public sector
- They need insight into the library

Use case #7: Hacker or competition

- There are non-customers and not-to-be-customers that wants to get insight into how a product is composed by getting the transparency docs.

The TEA discovery using a URN

https://github.com/CycloneDX/transparency-exchange-api/blob/main/discovery/readme.md

Existing product identifiers

- EAN code on marking
- App store vendor name and product name
- Product name and vendor
- SWID tag
- In app QR code (if possible) or identifier
- SKU - Stock Keeping Unit

Can we use these to find a unique identifier? Product names are rarely unique. Vendor names are not globally unique.

Including other IDENTIFIERS

- PURL - Package URL (soon an ECMA standard)
  - urn:tei:purl:
- EAN (Bar code identifier)
  - urn:tei:ean:
- GS1 bar codes
  - https://www.gs1.org/

How can we resolve these into a link to a document collection?

TEI: Transparency Exchange Identifier

- We need a unique identifier that all other identifiers can lead to
- An identifier that can be shown in the software itself, in MUD or be retrieved by other means
- This identifier needs to be a persistent identifier for a specific **product**, regardless of version
- The identifier needs to be globally unique - which is easiest to base on a **registered domain name.** Other solutions require a registry, a registrar and a registration process.

TEI

URN TEI name space: UUID

- **urn:tei:uuid:** for a company specific name and product identifier as UUID
  - `urn:tei:uuid:products.example.com:d4d9f54a-abcf-11ee-ac79-1a52914d44b1`
  - `urn:tei:uuid:<name based on domain>:<unique identifier>`
  - `urn:tei:uuid:products.example.com:d4d9f54a-abcf-11ee-ac79-1a52914d44b1?version=123.32`
- The name part does not have to exist as a web server. The uniqueness of the name is the domain part that has to be registred at creation of the TEI.
- The unique identifier has to be unique within the domain. Recommendation is to use UUID, but it can be a article code too.
- A TEI belongs to a single product. A product can have multiple TEIs.
  A product can consist of multiple parts, each with a single TEA index.

What do we do with the TEI?

- The DNS name part of the TEI is used in a DNS query to find one or multiple locations for product transparency exchange information.

- At the URL we need a well-known name space to find the API if there are no DNS records

```
urn:tei:uuid:products.example.com:d4d9f54a-abcf-11ee-ac79-1a52914d44b1
urn:tei:uuid:<name based on domain>:<unique identifier>
```

TEI

Transparency Product Index

- The first point of entry (TPI) is a list of available TVI (verson indexes) for various components (one or many)
- For each product, a structured document (TVI) contains a list of URIs on where to find the collection for **each supported version** of the product.
- The name of the structured document needs to be standardised and persistent in order to support automation.
- A document with no longer supported versions may need to exist as well.
    - *CRA mandates that documents as well as updates shall be available for all versions during the product lifetime*
- We need a redirect facility if products change ownership. (See lifecycle enumeration work in CycloneDX)

TEI DNS resolution

- The name in the DNS points to a set of DNS records
- TEI with name “tex.example.com" queries for _tei._tcp.tex.example.com URI records
- These point to URIs for the transparency exchange data
- If there are no records, try to resolve the name and use the /.well-known/tei prefix

```
_tei._tcp.tex.example.com.   3600 IN URI 10 1 “https://www.example.com/transparency“
```

DNS resolution to multiple URIs

```
_tei._tcp.tex.example.com.   3600 IN URI 10 1 “https://www.example.com/transparency“
_tei._tcp.tex.example.com.   3600 IN URI 20 1 “https://backup.example.com/transparency“
_tei._tcp.tex.example.com.   3600 IN URI 30 1 “https://thirdparty.example.org/example.com/transparency“
```

- First try lowest priority then move up.
- Recommended to have thirdparty external repositories as last priority
- The URI in DNS does not have to belong to the same domain as the URI records

Finding the index using dns

- Append the product part of the TEI to the URI found

TEI: urn:tei:uuid:**products.example.com**:d4d9f54a-abcf-11ee-ac79-1a52914d44b1

DNS record: _tei._tcp.products.example.com

URI in DNS: https://www.example.com/transparency/

URL: https://www.example.com/transparency/d4d9f54a-abcf-11ee-ac79-1a52914d44b1/

Finding the index using .well-known URI

- Append the product part of the TEI to the URI found
- Always prefix with the https:// scheme

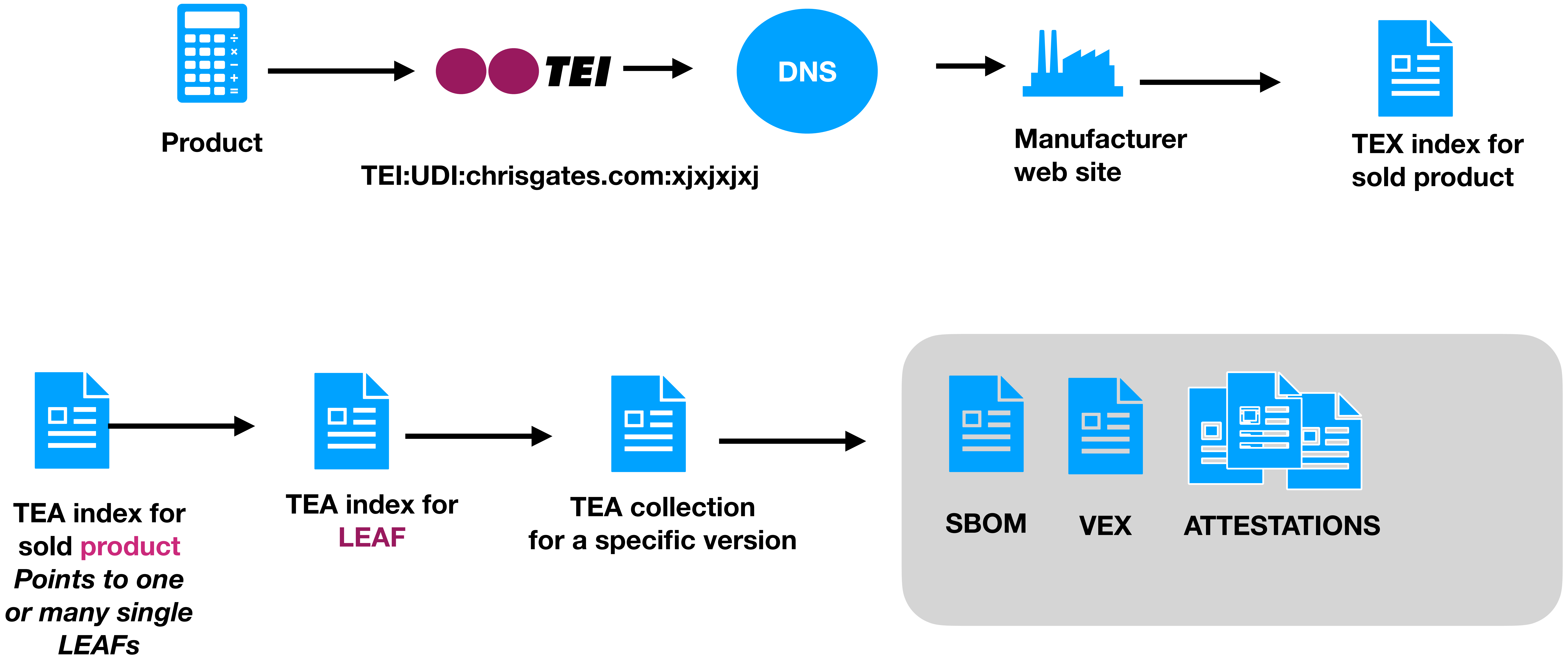
TEI: `urn:tei:uuid:products.example.com:`**`d4d9f54a-abcf-11ee-ac79-1a52914d44b1`**

URL: `https://products.example.com/.well-known/tei/`**`d4d9f54a-abcf-11ee-ac79-1a52914d44b1`**`/`

TEA Collections

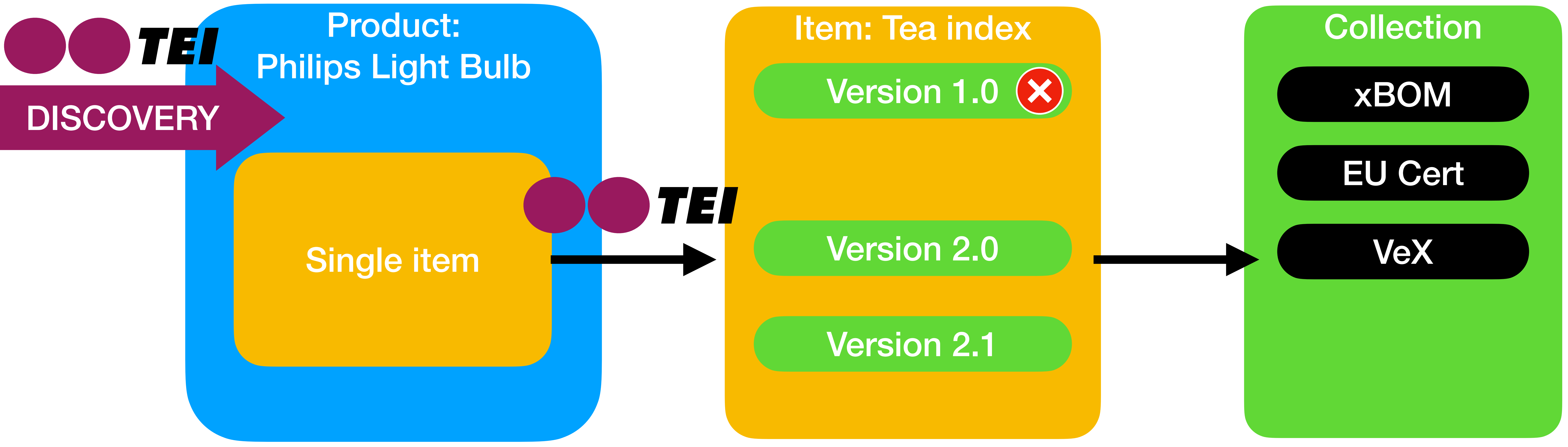
- The URL for a collection needs to be persistent to support automation
- A collection is a set of files (often signed) that applies to a specific product and version
- The collection index file indicates where to find SBOM, VEX, attestations and similar documents (like a conformance attestation for the CE marking in EU)

Discovery of transparency data


Product
TEI
TEI:UDI:chrisgates.com:xjxjxjxj
DNS
Manufacturer web site
TEX index for sold product
TEA index for sold product
Points to one or many single LEAFs
TEA index for LEAF
TEA collection for a specific version
SBOM
VEX
ATTESTATIONS

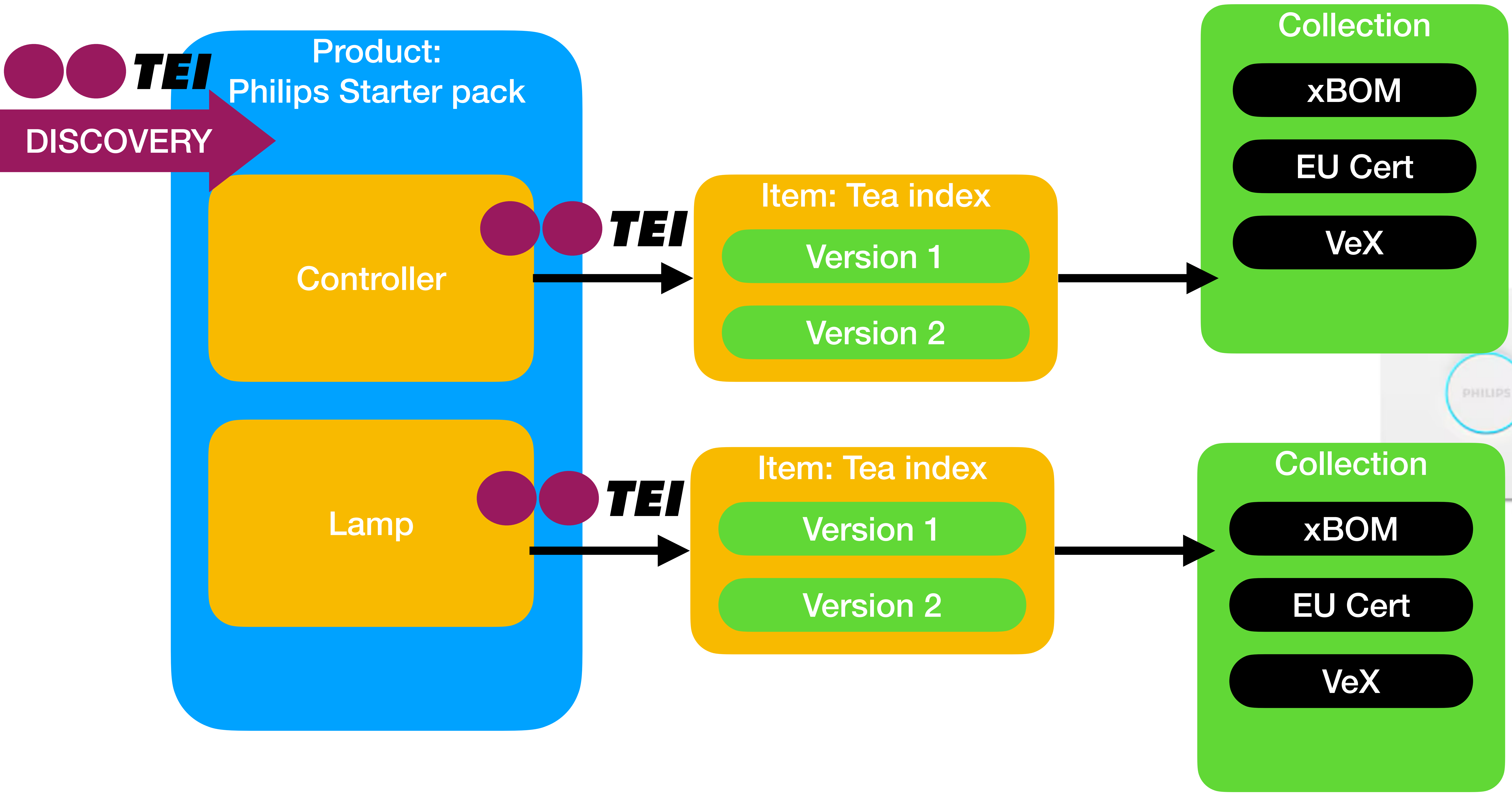
Examples on TPI usage

Light bulb

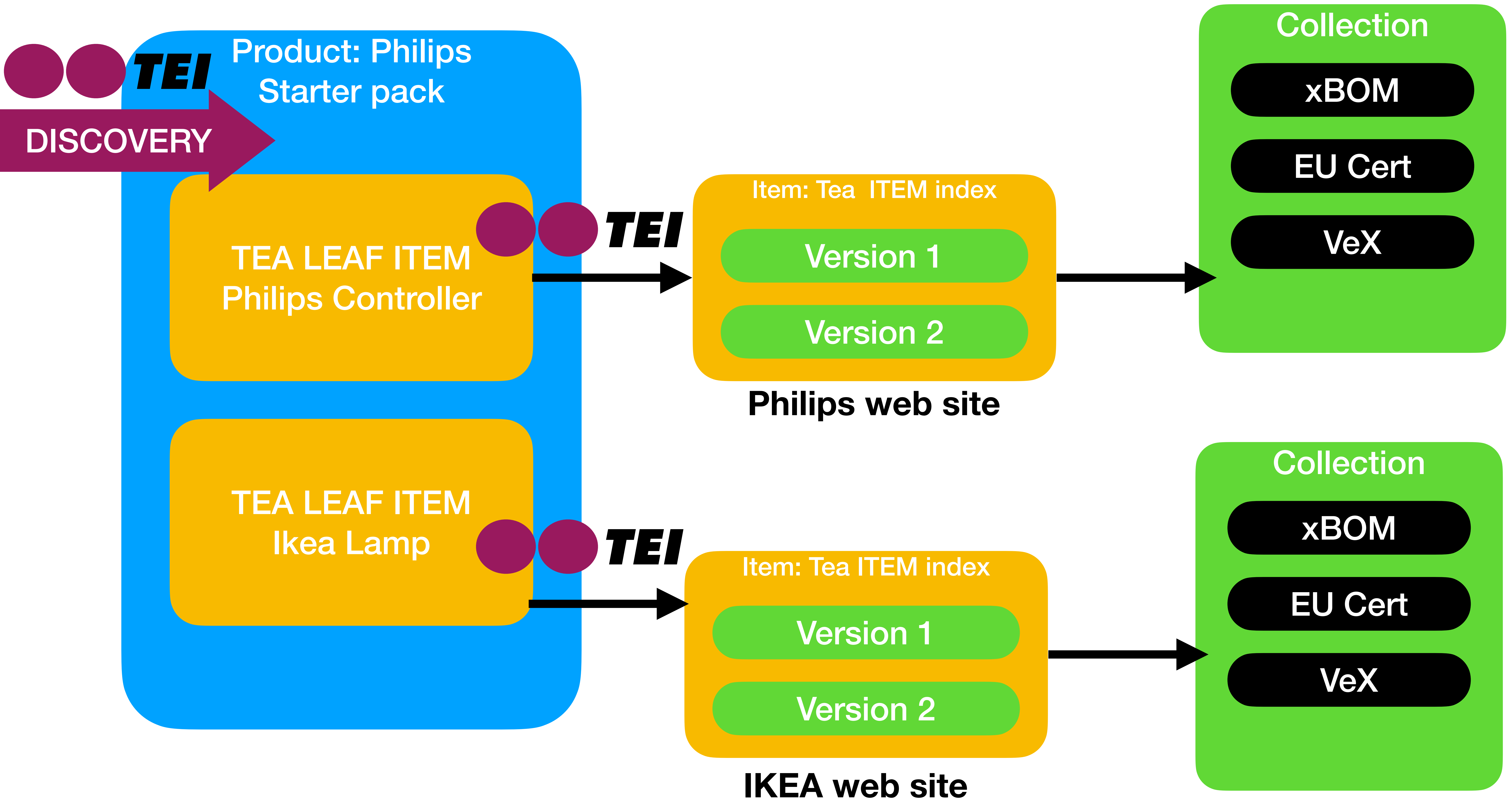

TEI
DISCOVERY
Product:
Philips Light Bulb
Single item
TEI
Item: Tea index
Version 1.0
Version 2.0
Version 2.1
Collection
xBOM
EU Cert
VeX

PHILIPS
hue
white and color
800

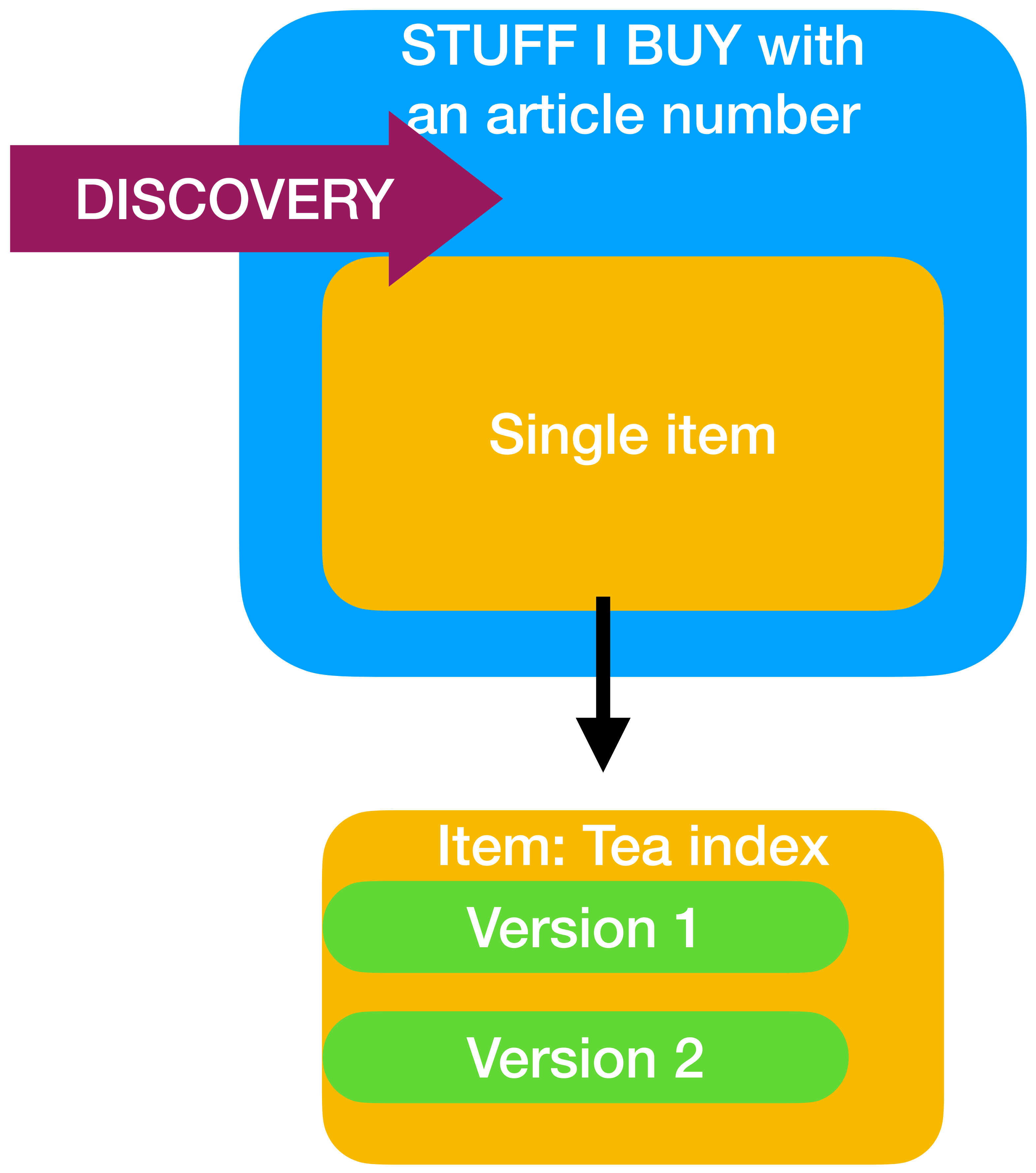
Starter pack

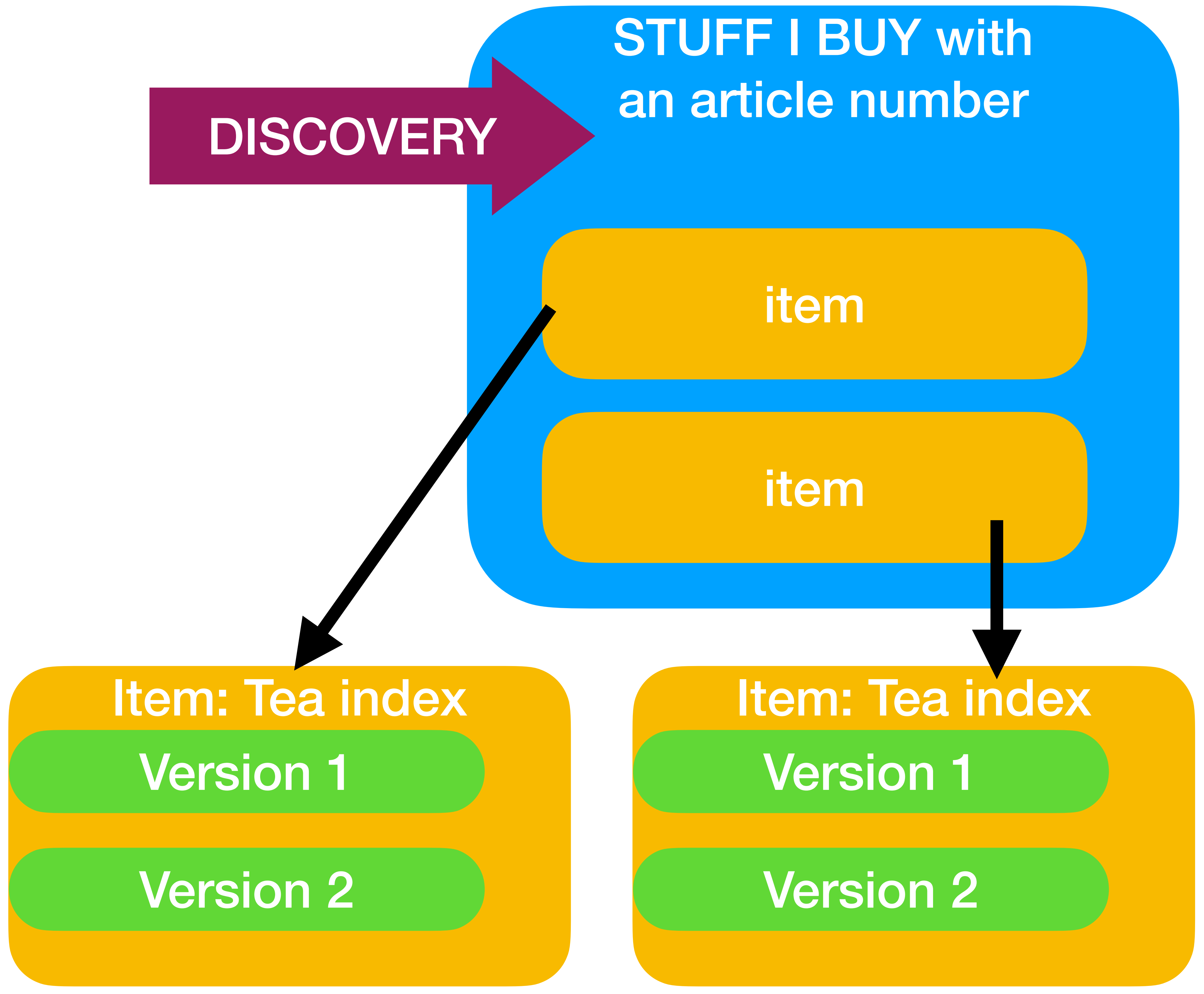


Pack from multiple vendors



Single item product vs multi item



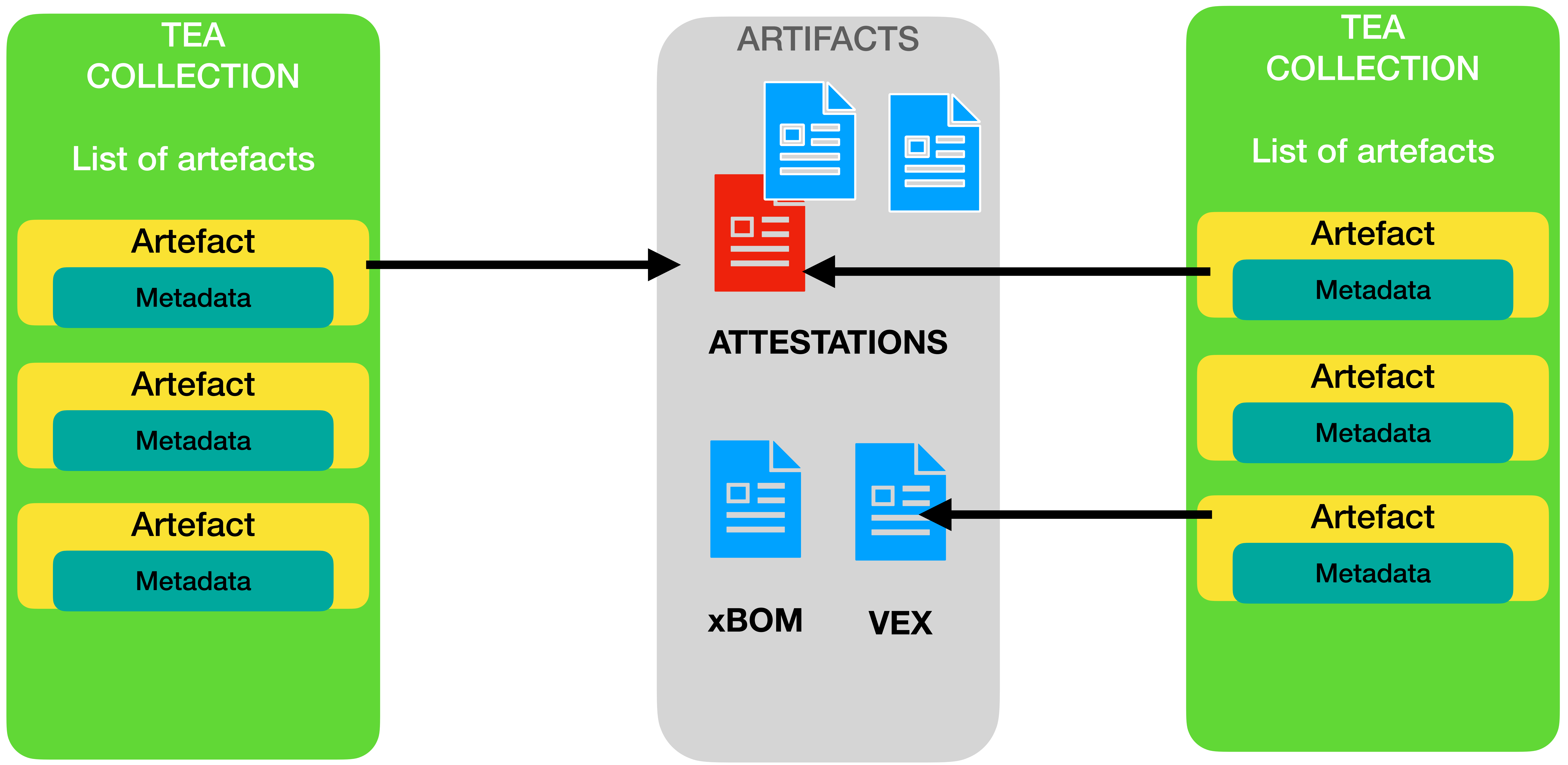


Proposal: TEA object model

TEI
DISCOVERY
TEA
PRODUCT INDEX
TEA LEAF ITEM
TPI
TEI
TEA LEAF INDEX
Philips Controller
Version 1
Version 2
Version 3
TLI
TEA
COLLECTION
List of artifacts
ARTIFACTS
xBOM
VEX
ATTESTATIONS

TEA PRODUCT INDEX
TEA LEAF ITEM
TPI
TEI
TEA LEAF INDEX
Philips Controller
Version 1
Version 2
Version 3
TLI
TEA COLLECTION
List of artifacts
TEA COLLECTION
List of artifacts
xBOM
VEX
ATTESTATIONS
One attestatio
in multiple col
ARTIFACTS
xBOM
VEX
ATTESTATIONS

The collection and artefacts



One artefact may belong to multiple collections.

TEI query options used within API

Option for versions

Option for “original product”

- urn:tei:uuid:products.example.com:d4d9f54a-abcf-11ee-ac79-1a52914d44b1
- urn:tei:uuid:products.example.com:d4d9f54a-abcf-11ee-ac79-1a52914d44b1?=version=3

Not documented in github
…yet…

Back to Alice - use case #1

- Alice aquires product in the store
- A QR code with the TEI can help if Alice has a vulnerability management software platform
- Otherwise a userfriendly URL can help her to the proper documents.
- Not the most user-friendly solution
- Can a third party public service help her with resolving the TEI and show the docs?

Use case #2

- A business consumer will get the TEI urn in shipping documentation, product documentation or within the product (if there’s a GUI)
- The TEI will be input in the vulnerability management system
- If the vendor disappears, the URI resolution will possibly point to a third party that keeps the documents during the product’s lifetime.
  - *A requirement here is that the base domain is still registered and operational to support DNS URI record resolution*
  - *If the vulnerability management system stores the latest DNS URI list, that dependency may not be a problem.*

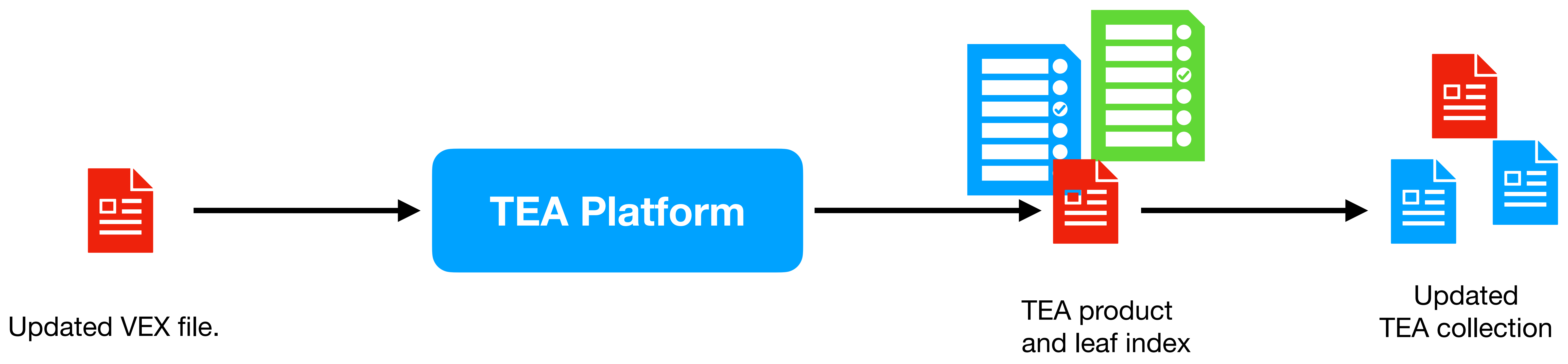
Authn/Authz

- We need to solve authentication and authorization in order to protect documents
- Authentication can stop competition and hackers
- But we need something easy to manage and persistent, so we don't have to spend a lot of time copy/pasting tokens or credentials between systems.
- Can we agree on one API mechanism for the API? Like a HTTP **bearer token**. How the token is aquired is out of scope. We may want to standardize how a system tells the API client that the token expired (if not already standardized)
- If the token is a signed JWT token (like in OpenIDconnect) authorization could be included in the token. **The Koala API doesn't need to specify the token content.**
- Scenario: I log in to Edvina support portal. The Edvina portal knows which products I have. I request an Koala API access token. Install that token in Dependency Track by copy/paste. The token is time limited.

TEA publishing

The transparency exchange API will support publication of signed artefacts.

A vendor or an open source project will be able to publish documents using the TEA API.



A global standard.

We’re part of ECMA TC54.

The **TEA API** is being developed as part of the ECMA TC54 working group in order to become an ECMA standard. TC54 is the Software and System transparency working group that standardise CycloneDX, PURL and the Transparency Exchange API.

TEA will be standardised in TG1 of ECMA TC54.

https://tc54.org/

Status update

Generic modules	Consumer	Publisher
Discovery: Stable	Open API spec: Alfa	Workflow: Started
Object model: Proposal	Security arch: Not yet	Open API spec: Not started
Use cases: Done	Authentication, authoritization Not documented	Security arch: Not yet
	Implementation: Not yet	Implementation: Not yet

Join the work!

We are working on writing specifications for the API and the various formats.

Join the OWASP CycloneDX Transparency Exchange API working group today to participate. We have a channel in the CycloneDX slack space to communicate.

https://github.com/CycloneDX/transparency-exchange-api

https://cyclonedx.org/about/participate/

OWASP

CycloneDX

Project Koala