

## eDAIRY APIs, versioning

---

**Author** : Conny Graumans, AgroConnect  
**Date** : 30jun22  
**Version** : v2022p01

---

### Document history:

- 30jun22: Copied from ePIGS Implementation Guideline. At its turn, inspired by ICAR implementation guideline. Source: Arjan Lamers.

-

---

## VERSIONING

Use Semantic Versioning 2.0.0 ( <https://semver.org/> ):

- change to **MAJOR** version indicates a breaking change
- change to **MINOR** version indicates a planned functional release (e.g. new message)
- change to **PATCH** version indicates a fix to the specification

Each change is a new version of the specifications. Although many types of changes do not result in a breaking change.

Only MAJOR version numbers are included in the URL, e.g.:

../edary-1/dairy\_delivery/en.ubn/1/deliveries

So, eDAIRY 1.0.0, 1.0.1, 1.23.46 all have the same URL

In terms of management of the standard: each MAJOR version has its own branch. Within it, every MINOR / PATCH version is tagged.

Each message type has its own endpoint (url). If the endpoint does not exist, the message is not supported by the data provider.

For data providers that means:

- To supporting a new message type, a new endpoint is published.
- A new MAJOR version means a new endpoint; this new endpoint can coexist with the old endpoint.

For data consumers this means:

- New message types never get in the way of old message types.
- So different message types can be supported in different MAJOR versions.

Note: new MINOR / PATCH versions will not get a new URL and need to be backwards compatible.

Each message type has a JSON schema definition. Fields can be added at any time.

For data providers this means:

- When new fields are added to a message, it is a MINOR version upgrade

For data consumers this means:

- Data consumers are supposed to use a tolerant reader pattern: ignore (new) fields that are not recognized.

Each message has a JSON schema definition. Each data field has a fixed definition that cannot be changed.

Changing definitions of data fields therefore means adding a new field.

For data providers that means:

- Both the old and the new meaning must be provided.
- When the old field is optional, it can be omitted. If it is mandatory, the old field will also have to be passed on until a MAJOR version upgrade.

For data consumers this means:

- Both the old and the new fields are available: the data consumer can choose when to switch from using the old data field in using the new data field.

Every data producer can add fields themselves, even if they are not defined in the standard. Do take into account the existing standards.

For data producers that means:

- You may add new non-standard data fields without notice.
- If the data field is potentially standardizable, define it so that it has the best chance of being adopted 1: 1: this will prevent changes later. Ideally, a MINOR version will be released later in which this new field is part of the standard
- If the field is vendor specific (or region specific,...) then use a prefix in the name of the field to avoid conflicting with future fields.

For data consumers this means:

- You are supposed to use a tolerant reader pattern: ignore fields you don't recognize.
- If you know the data producer and want to use the vendor-specific fields, you can. There are (from the standard) no guarantees about version control and quality.

-----  
-----