Compano Online Software

Compano Online Software

Manual JSON Feeds

Compano Online Software

Version 4.5

Document	Manual JSON Feeds
COS version	L04
Date	07/02/25

Content

1 Introduction

Data sharing with JSON Feeds

Compano Online Software JSON feeds are available for export of data to third parties and systems.

Some examples of useful applications of JSON feeds are:

Mono-lingual:

- Websites
- Webshops
- Catalogs

Multi-lingual:

- ERP systems
- CAD systems
- BIM systems

How does it work?

Feed lay-outs define which database fields and their values are exported. A feed lay-out is related to an entity, for example to *Products*. A Feed lay-out can contain a sub lay-out for several different related entities like *Items*. Thus Product data can be exported, including all related Item data.

Feeds can be configured to export data in one language or in multiple languages.

1.1 WARNING: Nightly downtime

Please, be mindful that Compano webservers are restarted each night. Restart times differ per application, but fall roughly between 1:00 and 6:00 AM. In case you need to know the specific downtime for your (client's) application, please contact Compano Support, support@compano.com.

2 Structure

A feed lay out is based upon a publication structure and can serve as an index for the related items and products. This will lead to a split feed:

- An index based upon the publication structure
- One or more end points for retrieving item and/or product data

In case there is no publication structure involved, the endpoints on item and/or product data can of course also be created and used. It is possible to filter and paginate certain data. It is basically possible to create endpoints for all available data in the PIM system, e.g. user defined fields.

3 Base URL

The base URL for each JSON feed:

```
https://{customer specific URL}/api/jsonfeed/{lay out name}
```

The URL in the example of this document is:

`https://pimtest.compano.com/API/jsonfeed/Products with related Items for webshop/1/1/?apikey=12398B7412642AB17647C823H64&filter=code=man001` and is composed of the following*:

The Base URL is `https://pimtest.compano.com/API/jsonfeed/Products with related Items for webshop/`

3.1 Feed parameters

3.1.1 Pagination parameters

Pagination parameters are added after the base URL, like thus: **`[Base URL]/1/1`**

3.1.2 Other parameters

The **first feed parameter** is added at the end of the base URL (with or without pagination), preceded by a question mark (?), for instance: **`[Base URL]?apikey=2C198773924B0834H3`**

Subsequent parameters are added after the first feed parameter, preceded by an ampersand (&) and separated by comma's (,)

For example:
`[Base URL]?apikey=2C198773924B0834H3&filter=Code=TAPCO,series=RKW,height=12,etc.`

4 API key

The API key is provided by Compano and is linked to a dedicated user account for web services, usually named 'webuser', in the PIM system. The API key will handle external JSON requests, and is passed to the Compano PIM system via the header of the request:

For example:

```
https://[redacted].compano.com/api/jsonfeed/[redacted]K_PRD/1/100?apikey=2C1[redacted]
[redacted]B1349¶
```

Note: The API key can be requested from Compano Support or your Compano consultant.

4.1 Number of calls

The number of concurrent calls that can be made to the Compano API is dependent on your user licence:

Standard user: Only one, single call
Concurrent user: Up to 50 concurrent calls

5 Pagination

After the base URL, the possibility exists to paginate the data, for instance 1/100/. The first number specifies the ***starting record***, the second number is a ***counter***, specifying the number of records per page, from the specified starting record onwards.

For example, starting at the 1st record, with 100 records per page:

```
https://{customer specific URL}/api/jsonfeed/{lay out
name}/1/100/
```

or, starting at the 101st record, with 50 records per page:

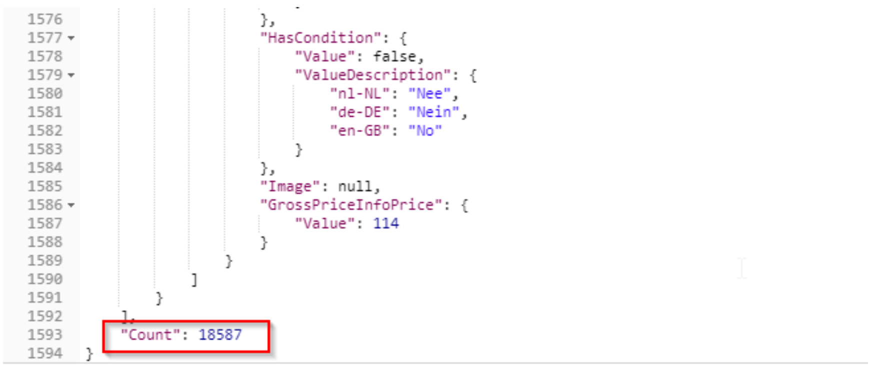
```
https://{customer specific URL}/api/jsonfeed/{lay out
name}/101/50/
```

To display **all records**, set the counter to `-1`:

```
https://{customer specific URL}/api/jsonfeed/{lay out name}/1/-1/
```

6 Count

At the end of each JSON output, a count of the number of records for that output is shown, for example:



```
                    },
                    "HasCondition": {
                        "Value": false,
                        "ValueDescription": {
                            "nl-NL": "Nee",
                            "de-DE": "Nein",
                            "en-GB": "No"
                        }
                    },
                    "Image": null,
                    "GrossPriceInfoPrice": {
                        "Value": 114
                    }
                }
            ]
        }
    ],
    "Count": 18587
}
```

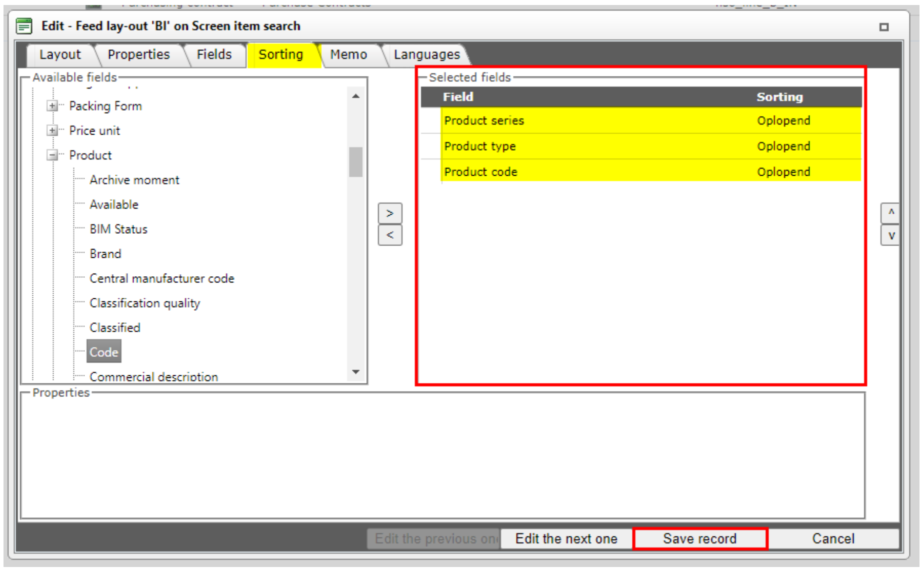
The ***count*** can make paginating easier: 'how many times do I need to paginate to show all records?' The base URL will give just a count of the number of records for the lay-out.

```
https://{customer specific URL}/api/jsonfeed/{lay out name}/
```

7 Sorting

By design, the JSON feed output is *unsorted*. Sorting can be done in the destination website or system.

As of COS version L03, sorting of the feed can now be set by adding fields on the sorting tab of the feed layout:



Edit - Feed lay-out 'BI' on Screen item search

Layout | Properties | Fields | Sorting | Memo | Languages

Available fields
- Packing Form
- Price unit
- Product
  - Archive moment
  - Available
  - BIM Status
  - Brand
  - Central manufacturer code
  - Classification quality
  - Classified
  - Code
  - Commercial description

Selected fields

Field	Sorting
Product series	Oplopend
Product type	Oplopend
Product code	Oplopend

Properties

Edit the previous one | Edit the next one | Save record | Cancel

- Sorting can be set to either *Ascending* or *Descending*
- Sorting on multiple data fields is hierarchical; in the example above the products will be sorted first on Series, secondly within Series on Type and finally within Type on Product Code.

<u>Note</u>: Sorting is only valid for the ***main*** feed; any associated sub-feeds will <u>not</u> be sorted according to the settings of the main feed.

8 Filters

Several filters can be applied to a JSON feed.

8.1 Filter separator

In calling a JSON feed, the default Filter Separator is a ***colon*** character. However, an alternative Filter Separator can be defined by calling the JSON feed with the following option:
`filterseparator=[character]`

For example:
[CustomerURL]/api/jsonfeed/{lay out name}/1/100/?filter=ProductCode=4,210,050,001;Code=241015&filterseparator=;

8.2 On database field

The parameter to filter on a ***database field*** is: `?filter={db field}={value}`

For example:
```
https://{customer specific URL}/api/jsonfeed/{lay out
name}/1/100/?filter={db field1}={value}
```

or multiple filters:
```
https://{customer specific URL}/api/jsonfeed/{lay out
name}/1/100/?filter={db field1}={value},{db field2}={value},{db
field3}={value}
```

8.3 Datesince=

By using ***Datesince*** it is possible to filter data on a certain date. The *Datesince* function will filter products or items that have been altered/created *on* or *after* the provided value of *Datesince*. The format of the *date* in the *Datesince* function is always: `YYYYMMDD`

For example[1]:
`?datesince=20170703`
or
`&datesince=20170703`

NOTE!: For *Items* and *Products* the field ***Modification Time*** *(Overall)* is used: `CompositeLastModificationDateTime`

For ***Items*** (`Item.CompositeLastModificationDateTime`) this includes changes in related fields, such as:

- Attachments (`Attachment`)

[1] The first filter parameter is added to the URL with the prefix `?`, any subsequent parameters are added using the prefix `&`. Within a filter different fieldname=value code is separated by a comma `,` For example:
```
https://{customer specific URL}/api/jsonfeed/{layout
name}/1/100?filter=ManufacturerCode=TAPCO,series=RKW,height=12&datesince=20
181201
```

- Price Information (`PriceInfo`)
- Surcharges (`ItemSurcharge`)
- Purchasing Conditions (`ActivePurchaseCondition`)
- Accessory Products (`AddOn`)
- Product Modification Time (Overall) (`Product.CompositeLastModificationDateTime`)
- Group Code (`ItemGroup`)
- Supplier (`SalesOrganization`)

For ***Products*** (`Product.CompositeLastModificationDateTime`) this includes changes in the related fields:

- Attachments (`Attachment`)
- Manufacturer (`Manufacturer`)
- Accessory products (`AddOn`)
- Product parts (`SubProduct`)
- Group Code (`ProductGroup`)

For ***all other entities*** the field *Modification Time* is used: `LastModificationDateTime`

8.4 =IsNewForFeed=

The combination of the parameters *&Datesince=* and *IsNewForFeed* adds the possibility to discern whether a product has been created or whether it has been altered on or after the *DateSince*.

In the JSON output the *IsNewForFeed* field can have two values:

- `True`: The product/item is new
- `False`: The product/item has been altered.

IsNewForFeed does not function when an item or product feed is nested under another feed like a feed based upon a publication structure.

To filter on *IsNewForFeed* adds the following to the URL:

```
?filter=Isnewforfeed={true/false}
```

or

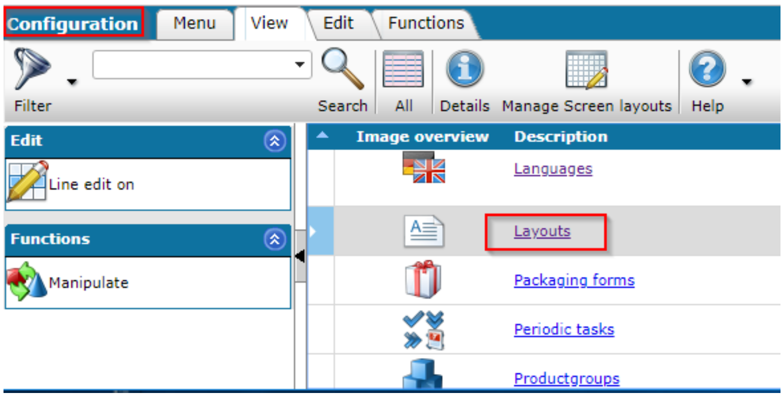
```
&filter=Isnewforfeed={true/false}
```

9 Create feed layouts

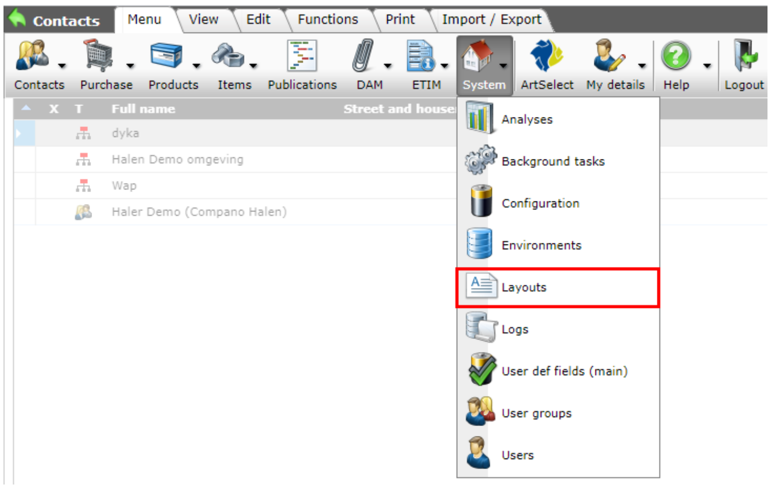
The first step in implementing a JSON Feed is to create a ***Feed Layout***. Feed layouts are used to select the database field values which are to be exported. A layout is related to a main entity, for example: *Products*. A Feed layout can contain a ***sub-layout*** for a related entity, for instance *Items*. This enables you to export Products with their Items.

In the explanation below, a main and sub-feed layout is created for exporting products with their related items for a web shop:

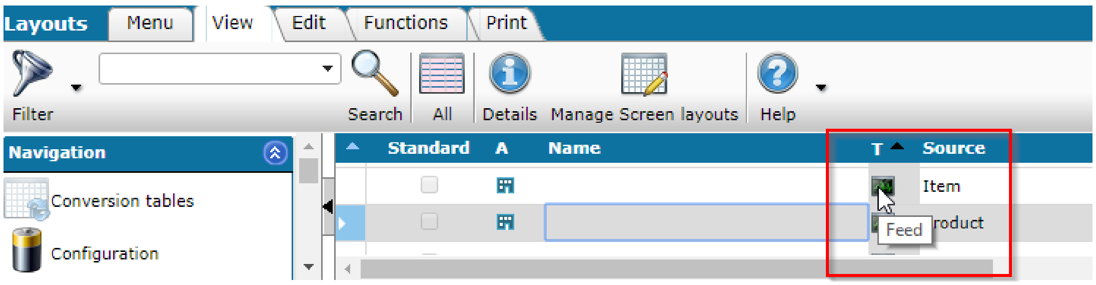
1. Through the Menu go to *System > Configuration > Layouts* and click on *Layouts*. Alternatively, as of COS version L03, go to *System > Layouts*.



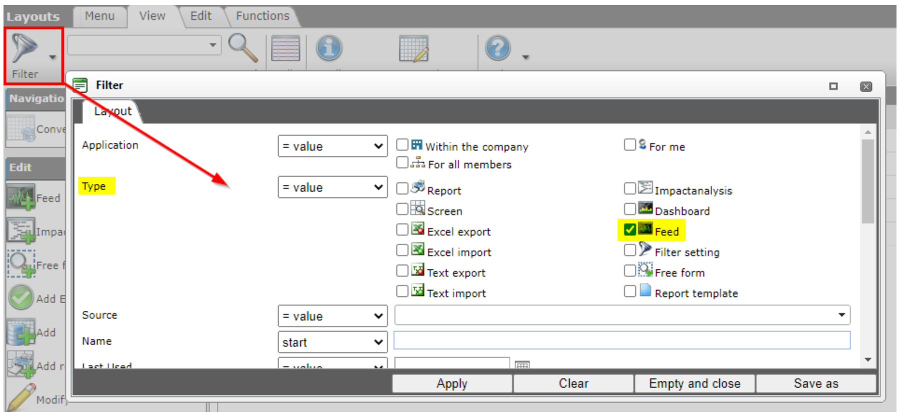
Or, as of COS version L03:



2. Layouts are listed by Type; a feed layout can be recognized by the icon :



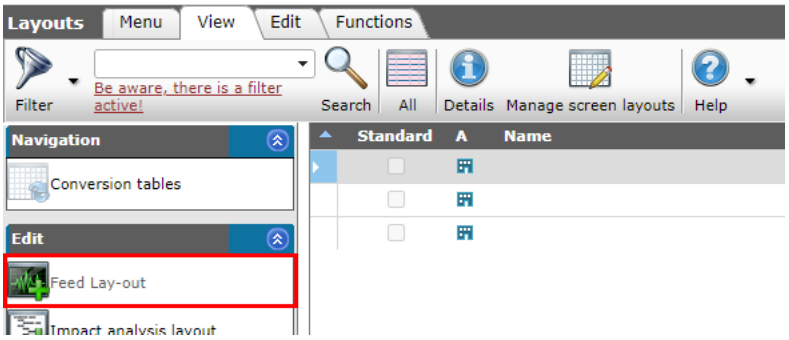
   a. <u>Note</u>: Feed layouts can also be filtered out using the Comprehensive Filter:



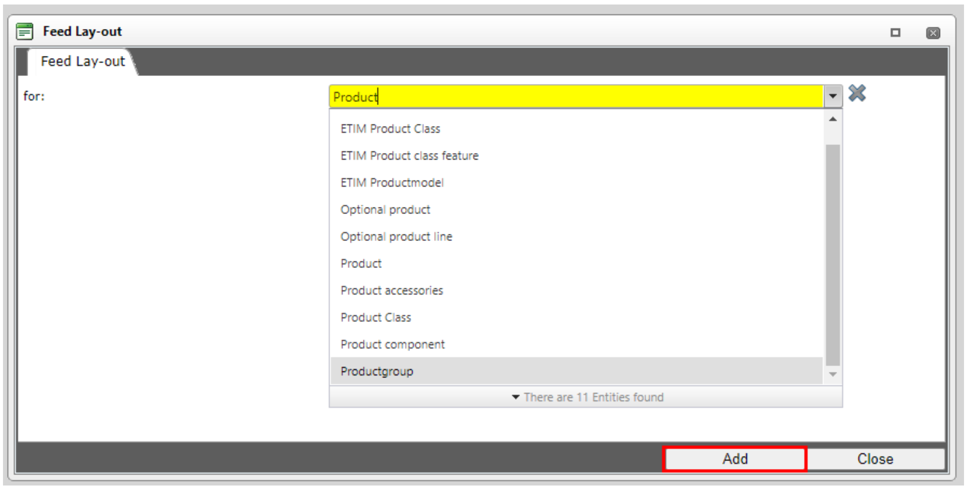
Filter

Application	= value	☐ Within the company ☐ For all members	☐ For me
Type	= value	☐ Report ☐ Screen ☐ Excel export ☐ Excel import ☐ Text export ☐ Text import	☐ Impactanalysis ☐ Dashboard ☑ Feed ☐ Filter setting ☐ Free form ☐ Report template
Source	= value		
Name	start		
Last Used	= value		

3. To create a new feed layout, under *Edit* click on *+Feed Layout:*



4. On the next screen, choose which entity to create the feed for and click on Add.

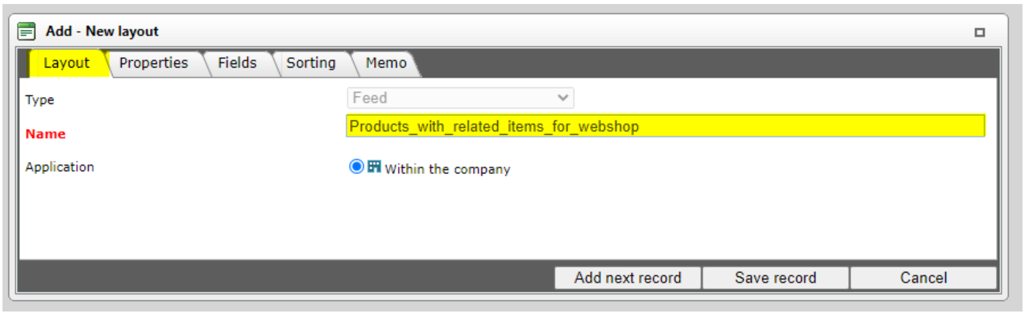


for: Product

ETIM Product Class
ETIM Product class feature
ETIM Productmodel
Optional product
Optional product line
Product
Product accessories
Product Class
Product component
Productgroup

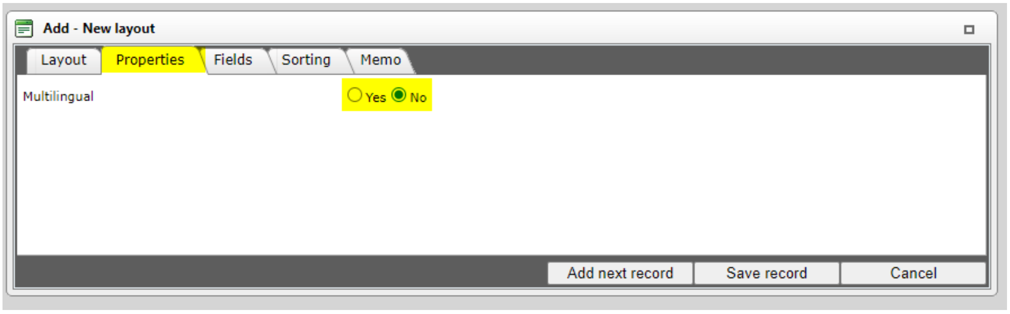
There are 11 Entities found

5. On the *Layout* tab, enter a unique name for the feed. Note: It is recommended to replace spaces with either a hyphen or underscore character, as web application sometimes do not handle spaces in names correctly:



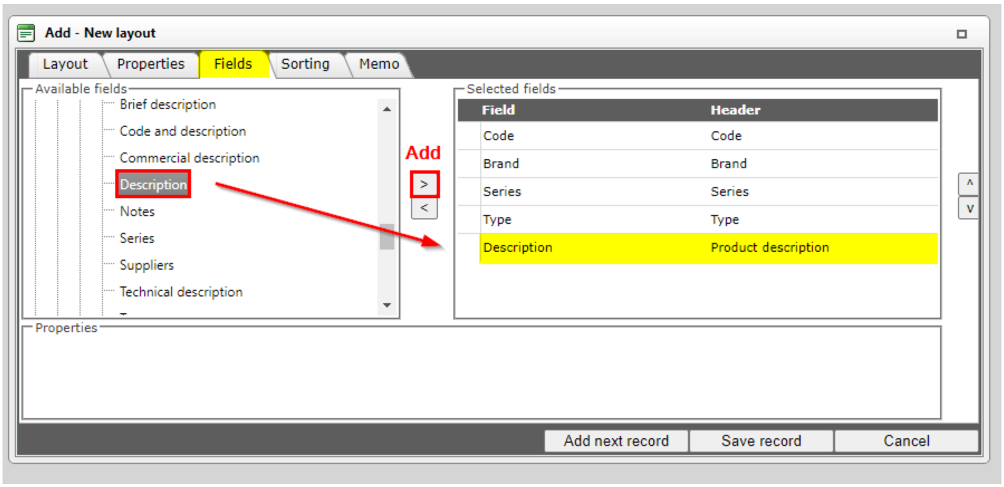
Type: Feed
Name: Products_with_related_items_for_webshop
Application: Within the company

6. On the *Properties* tab, set Multilingual to *Yes* or *No*. This option will be explained in further detail in chapter *12 Language scenarios*.



Multilingual: Yes [✓ No]

7. On the *Fields* tab, start construction your feed by selecting and adding database fields. The descriptions and values of these fields will be exported when the feed is called.
   a. Select a field under *Available fields* and use the arrows or 'double-click' to add it to *Selected fields*.

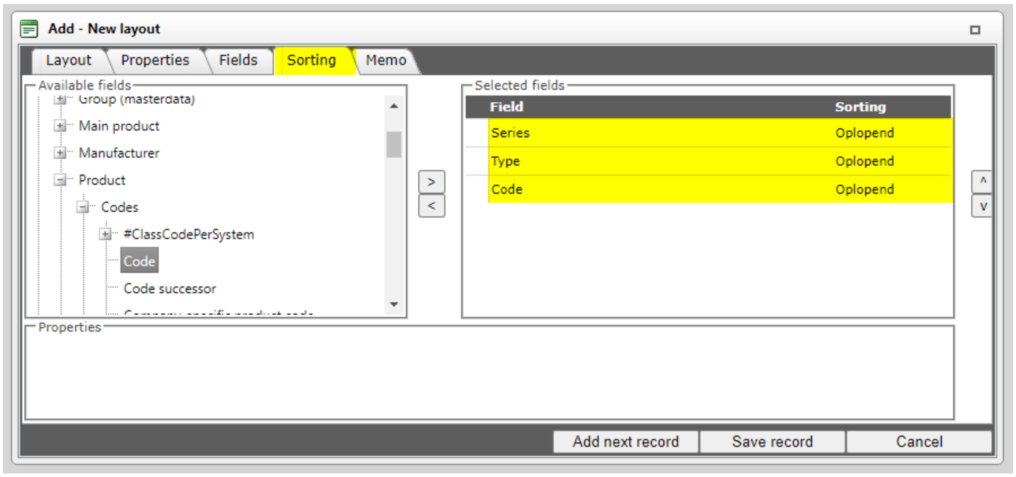


Available fields:
- Brief description
- Code and description
- Commercial description
- Description
- Notes
- Series
- Suppliers
- Technical description

Selected fields:

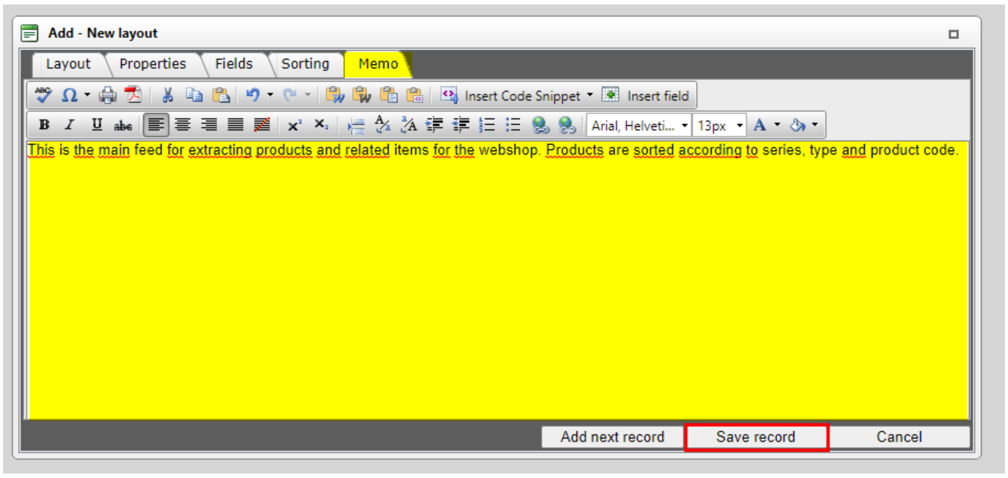
Field	Header
Code	Code
Brand	Brand
Series	Series
Type	Type
Description	Product description

8. On the *Sorting* tab, optionally add any fields on which the feed needs to be sorted. For a detailed explanation, see chapter *7 Sorting*.



Field	Sorting
Series	Oplopend
Type	Oplopend
Code	Oplopend

9. On the *Memo* tab, optionally enter any additional information regarding this feed. This tab can be used to take notes on why and how this feed was made. <u>Note</u>: This information will <u>not</u> be exported by the feed.



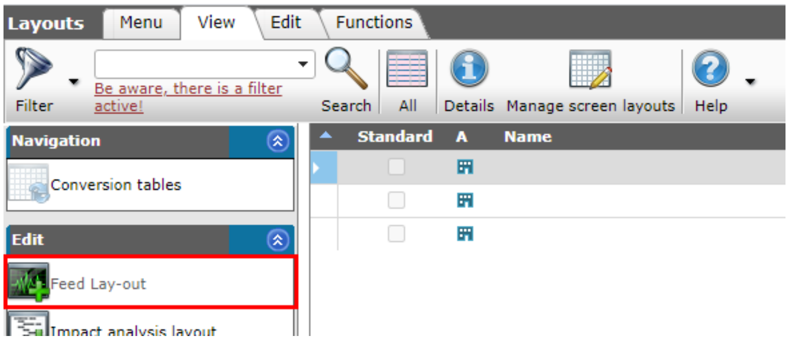
This is the main feed for extracting products and related items for the webshop. Products are sorted according to series, type and product code.

10. When ready, *Save* the feed.

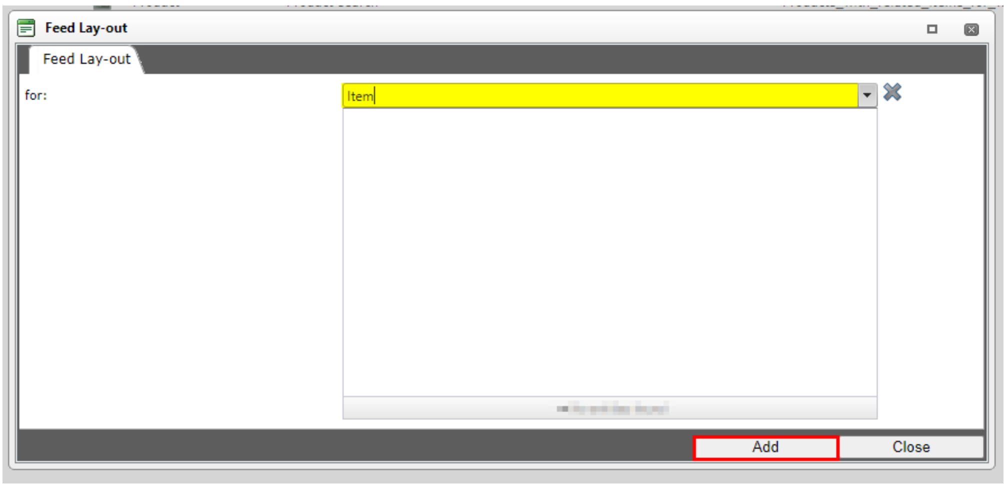
9.1 Adding a sub-feed

Now, to add related Item information to the Product feed of the previous paragraph, create a ***second*** (sub-)feed for the entity Item:

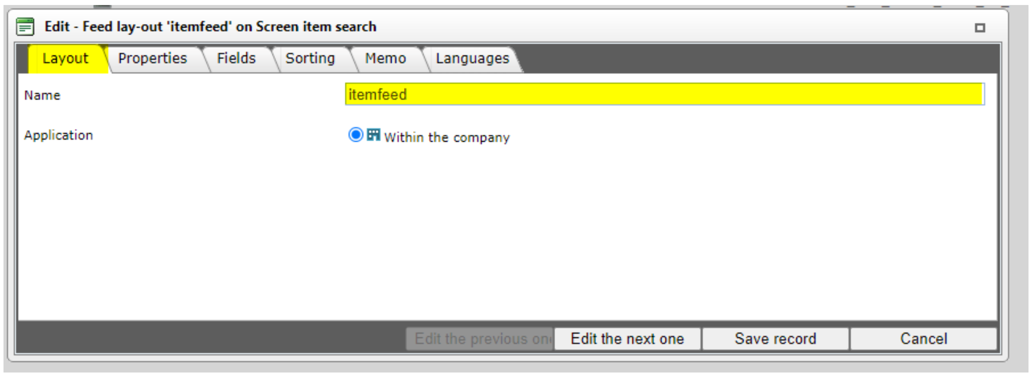
1. Again, under Edit, click on *+Feed Layout* to create a sub-feed layout:



2. Next, choose the entity *Item*.



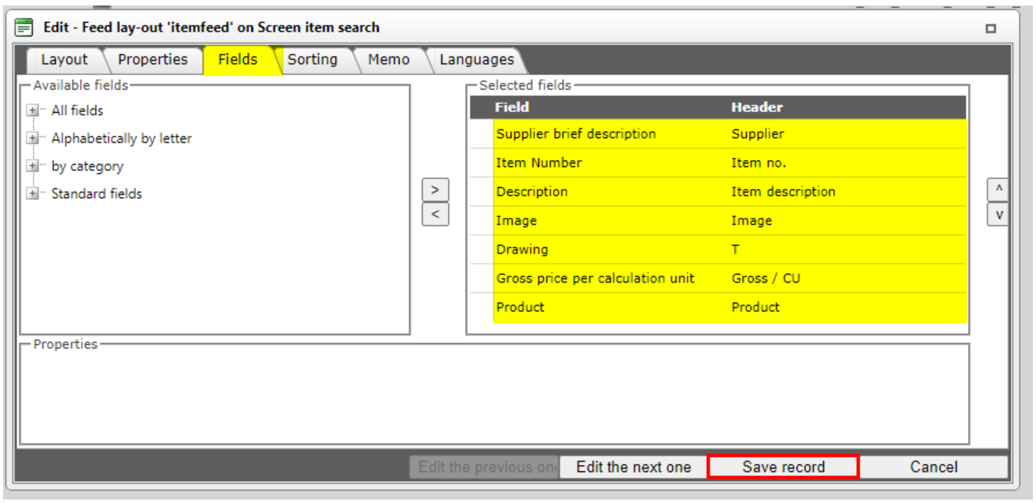
3. On the *Layout* tab, enter a unique name. Remember not to avoid using spaces.



Name: itemfeed

Application: Within the company

4. On the Fields tab, select and add Item fields that need to be included in the feed, for instance Item Number, Gross Price, etc.:

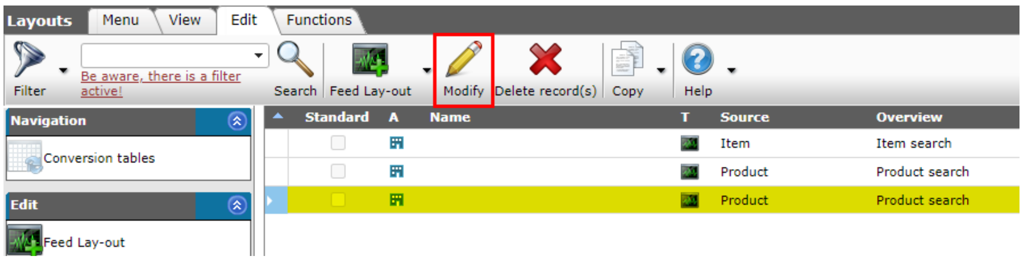


Field	Header
Supplier brief description	Supplier
Item Number	Item no.
Description	Item description
Image	Image
Drawing	T
Gross price per calculation unit	Gross / CU
Product	Product

5. Save the feed by clicking on *Save record*.

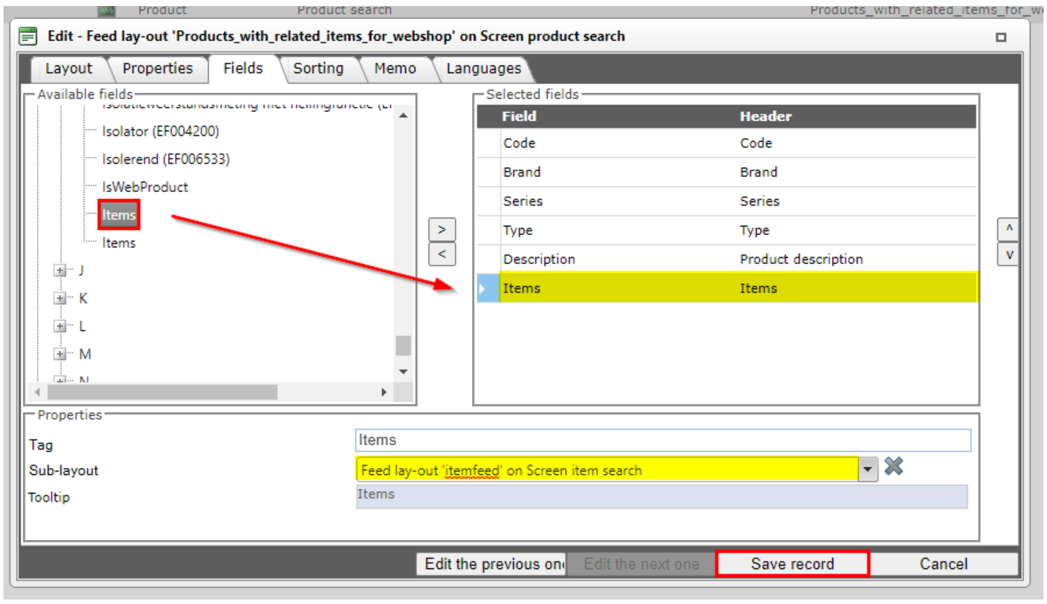
Next, add the Item feed layout as a sub-layout to the Product feed:

1. Select the Product feed layout in the layout overview and, under *Edit*, click on *Modify*.



Standard	A	Name	T	Source	Overview
				Item	Item search
				Product	Product search
				Product	Product search

2. Go to the *Fields* tab and add the field *Items* to the Product feed layout:


Edit - Feed lay-out 'Products_with_related_items_for_webshop' on Screen product search

Field	Header
Code	Code
Brand	Brand
Series	Series
Type	Type
Description	Product description
Items	Items

Properties

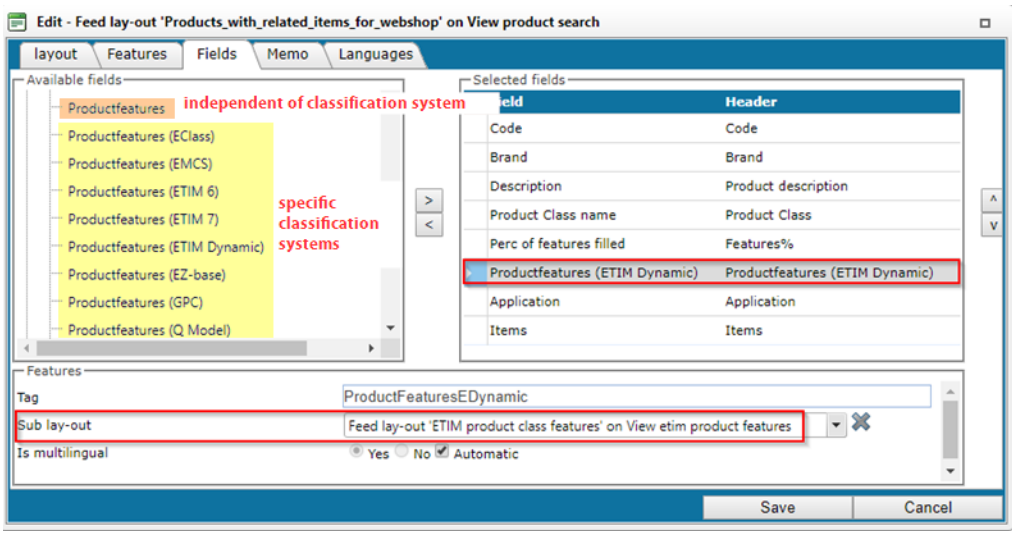
Tag	Items
Sub-layout	Feed lay-out 'itemfeed' on Screen item search
Tooltip	Items

3. Under *Selected fields*, select the *Items* field and enter the following Features:
   a. **Tag**: Optionally change the Tag text.
   b. **Sub-layout**: Select the Item feed you made previously from the drop-down menu.
   c. **Tooltip**: Optionally add a tooltip text.
4. Save the feed by clicking on *Save record*.

9.2 Adding Product Classification to your JSON feed layout

In the case that products have been classified in one specific or multiple classifications systems, then you would have to add the field Productfeatures to your product feed layout. Then create a separate (sub) layout for the entity *Product features* and link it to the field Productfeatures to include them in your feed.

Note: The contents of "productfeatures" and "productmodelfeatures" (in all its variants) have a FIXED layout, that you cannot influence. But, since the system will check on the existence of a sub layout you DO have to define at least one (dummy) layout for (ETIM) product features:


Edit - Feed lay-out 'Products_with_related_items_for_webshop' on View product search
layout
Features
Fields
Memo
Languages
Available fields
independent of classification system
specific classification systems
Selected fields
Save
Cancel

Available fields:

- Productfeatures
- Productfeatures (EClass)
- Productfeatures (EMCS)
- Productfeatures (ETIM 6)
- Productfeatures (ETIM 7)
- Productfeatures (ETIM Dynamic)
- Productfeatures (EZ-base)
- Productfeatures (GPC)
- Productfeatures (Q Model)

eld	Header
Code	Code
Brand	Brand
Description	Product description
Product Class name	Product Class
Perc of features filled	Features%
Productfeatures (ETIM Dynamic)	Productfeatures (ETIM Dynamic)
Application	Application
Items	Items

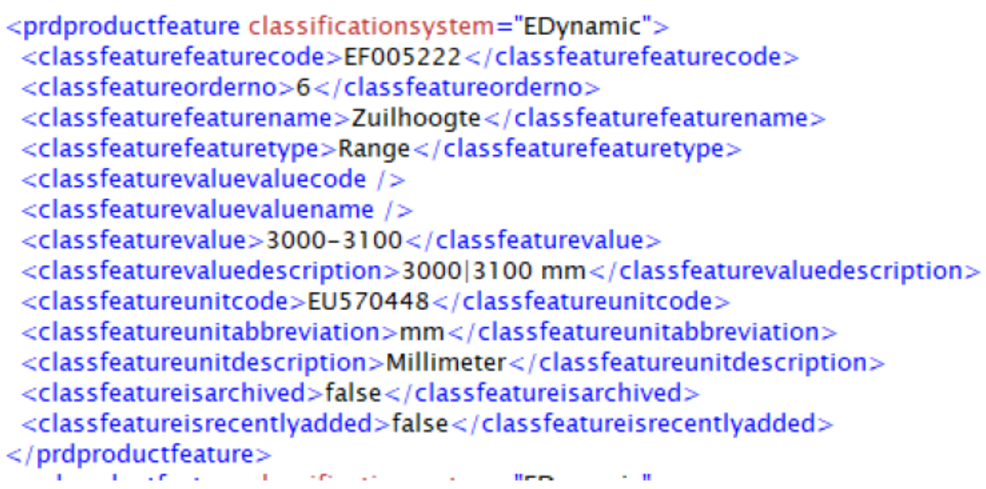
Features

Tag	ProductFeaturesEDynamic
Sub lay-out	Feed lay-out 'ETIM product class features' on View etim product features
Is multilingual	Yes No [✓ Automatic]

The *Productfeatures* tag can be added as a 'classification system neutral' tag or as a tag that is specific to one classification system, such as *Productfeatures (EZ-base)* or *Productfeatures (Q model)*, etc.

<u>Notes</u>:

1. Use specific tags only if creating a JSON feed. For XML feeds, only the independent tag can be used.
2. If multiple *Productfeature* tags (of different systems) have been added, please notify the company that will process the feed that per product multiple feature sets will be present in the feed.
3. Content of the *Productfeature* tag is hard-coded:
   a. For the XML variant the *Productfeature* tag consists of 13 sub tags:



```
<prdproductfeature classificationsystem="EDynamic">
  <classfeaturefeaturecode>EF005222</classfeaturefeaturecode>
  <classfeatureorderno>6</classfeatureorderno>
  <classfeaturefeaturename>Zuilhoogte</classfeaturefeaturename>
  <classfeaturefeaturetype>Range</classfeaturefeaturetype>
  <classfeaturevaluevaluecode />
  <classfeaturevaluevaluename />
  <classfeaturevalue>3000-3100</classfeaturevalue>
  <classfeaturevaluedescription>3000|3100 mm</classfeaturevaluedescription>
  <classfeatureunitcode>EU570448</classfeatureunitcode>
  <classfeatureunitabbreviation>mm</classfeatureunitabbreviation>
  <classfeatureunitdescription>Millimeter</classfeatureunitdescription>
  <classfeatureisarchived>false</classfeatureisarchived>
  <classfeatureisrecentlyadded>false</classfeatureisrecentlyadded>
</prdproductfeature>
```

   b. For the JSON variant the content and number of tags varies per type of *Productfeature* chosen.
   c. For the JSON variant you must choose a layout for the *Productfeature* sub tag, however content for this layout is hard-coded.

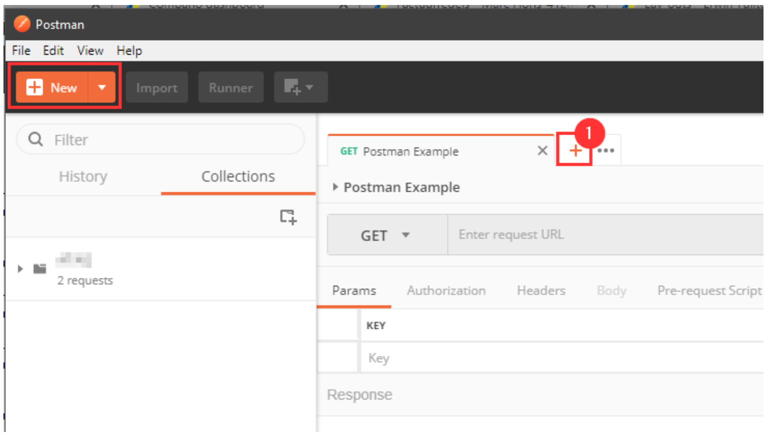
10 Test Feed Lay-out with Postman

The feed can be tested and edited with the free application ***Postman***:

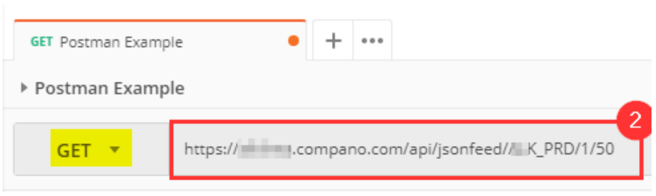
https://www.getpostman.com/apps

To test a feed:

1. Click on the plus sign (1) to add a new *Get Request:*

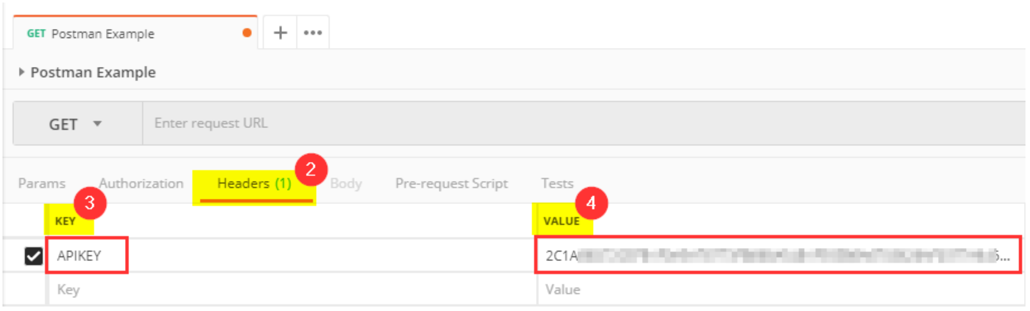


2. Next to GET (2) enter the Request URL, including any pagination parameters:



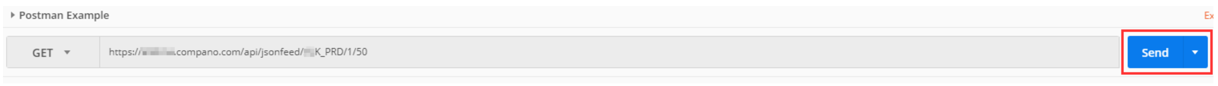
https://[redacted].compano.com/api/jsonfeed/[redacted]K_PRD/1/50

3. Go to the tab *Headers* (3) and under KEY (4) enter `APIKEY`:



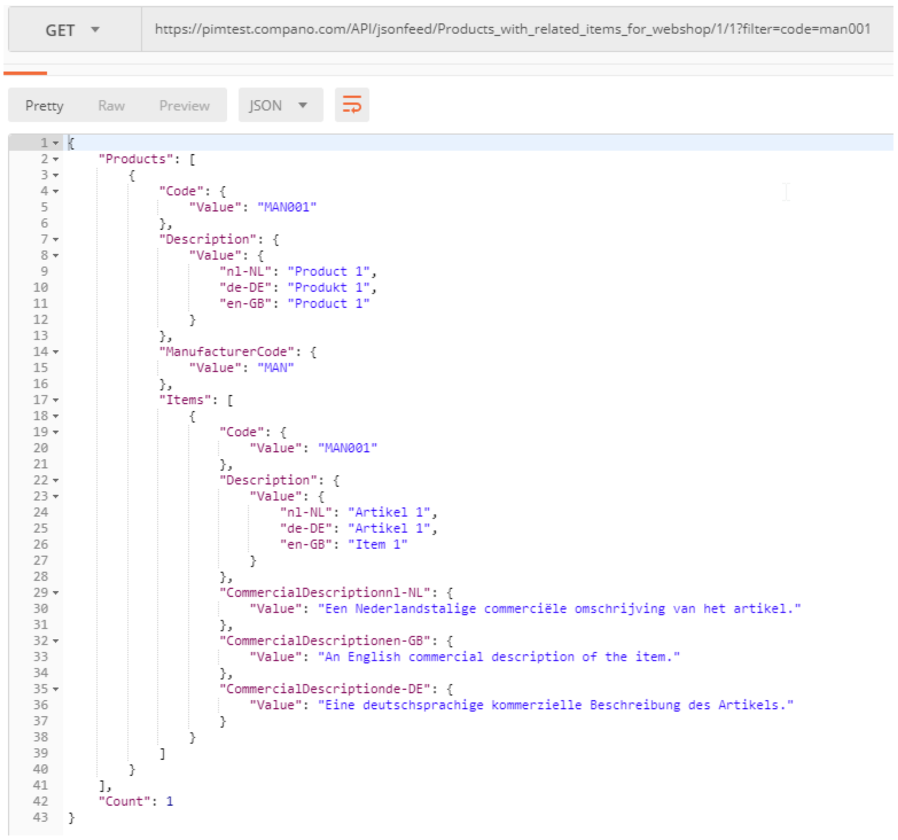
KEY	VALUE
APIKEY	2C1A[redacted]...

4. Under VALUE (5), enter the ***API key*** of the 'webservices' user (see chapter *4 API key*)



https://[redacted].compano.com/api/jsonfeed/[redacted]K_PRD/1/50

5. Click *Send* to get (the results of) the JSON Feed URL. Below is a very simple example of a multilingual feed for just one product with code MAN001 with its related articles.



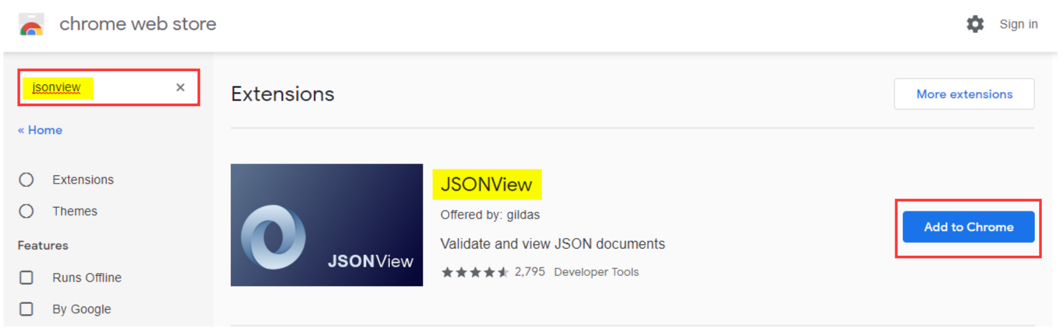
https://pimtest.compano.com/API/jsonfeed/Products_with_related_items_for_webshop/1/1?filter=code=man001

```
{
    "Products": [
        {
            "Code": {
                "Value": "MAN001"
            },
            "Description": {
                "Value": {
                    "nl-NL": "Product 1",
                    "de-DE": "Produkt 1",
                    "en-GB": "Product 1"
                }
            },
            "ManufacturerCode": {
                "Value": "MAN"
            },
            "Items": [
                {
                    "Code": {
                        "Value": "MAN001"
                    },
                    "Description": {
                        "Value": {
                            "nl-NL": "Artikel 1",
                            "de-DE": "Artikel 1",
                            "en-GB": "Item 1"
                        }
                    },
                    "CommercialDescriptionnl-NL": {
                        "Value": "Een Nederlandstalige commerciële omschrijving van het artikel."
                    },
                    "CommercialDescriptionen-GB": {
                        "Value": "An English commercial description of the item."
                    },
                    "CommercialDescriptionde-DE": {
                        "Value": "Eine deutschsprachige kommerzielle Beschreibung des Artikels."
                    }
                }
            ]
        }
    ],
    "Count": 1
}
```

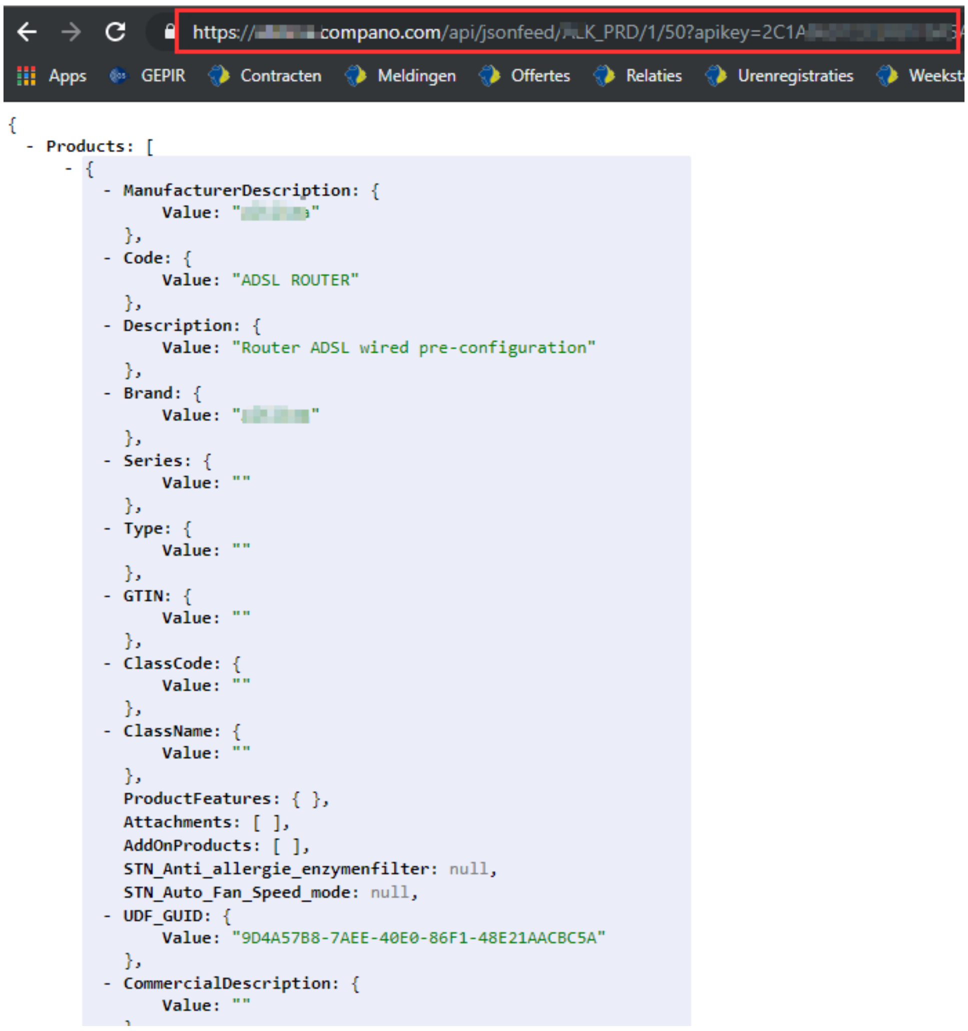
11 View JSON feed in Google Chrome

To view a JSON feed in the Google Chrome browser, use a browser extension like *JSONView*.

1. Download the ***JSONView*** extension from: https://chrome.google.com/webstore/



2. Add the extension to Chrome
3. Type the JSON feed URL, including *apikey* Chrome's address bar. Results will now be shown in a structured lay-out. Using the plus- and minus-signs, you can extend or collaps parts of the feed:


https://[redacted]compano.com/api/jsonfeed/[redacted]LK_PRD/1/50?apikey=2C1A[redacted]
Apps
GEPIR
Contracten
Meldingen
Offertes
Relaties
Urenregistraties
Weekst

```
{
  - Products: [
      - {
          - ManufacturerDescription: {
                Value: "[redacted]"
            },
          - Code: {
                Value: "ADSL ROUTER"
            },
          - Description: {
                Value: "Router ADSL wired pre-configuration"
            },
          - Brand: {
                Value: "[redacted]"
            },
          - Series: {
                Value: ""
            },
          - Type: {
                Value: ""
            },
          - GTIN: {
                Value: ""
            },
          - ClassCode: {
                Value: ""
            },
          - ClassName: {
                Value: ""
            },
            ProductFeatures: { },
            Attachments: [ ],
            AddOnProducts: [ ],
            STN_Anti_allergie_enzymenfilter: null,
            STN_Auto_Fan_Speed_mode: null,
          - UDF_GUID: {
                Value: "9D4A57B8-7AEE-40E0-86F1-48E21AACBC5A"
            },
          - CommercialDescription: {
                Value: ""
```

12 Language scenarios

Languages in Compano Software are ordered hierarchically:

- (Feed) lay-out language
    - User language
        - System language

There are 4 basic configurations for the hierarchy for languages in the JSON feed:

1. Define only one language by adding the filter *&culture=* to the URL
2. Define multiple languages by adding them to the feed lay-out
3. User language of the user that runs the feed
4. Database language for the description of multilingual text fields that are marked as 'No' for multilingual

See below a table for scenarios:

	Customer multilingual	**Customer monolingual**
JSON Feed multilingual	*-For system databases-* Use **Lay-out languages**. But use **Database language** for exceptions at field level (of type multilingual description), like brand.	n/a
JSON Feed monolingual	*-For websites-* Use filter **&culture=[ISO 639-1 language code combined with ISO-3166-1 alpha-2 standaard]**[2]	*-For system databases-* Use **User language.** Option 'Add Value Description' for field types: - Enum (single select) - Multiple Select - Boolean (True/False) - For multilingual text fields where Multilingual is set as 'Yes'

12.1 Configuring monolingual Feeds

12.1.1 &culture=

A multilingual customer can generate a *Monolingual* feed for a website by using the filter **&culture=** followed by the language ISO code.

Any languages selected on the tab Languages and multilingual settings (yes or no) at lay-out field level are ignored.

In the following example URL the only language is English, so the filter will be followed by the ISO code *en-GB*. So the URL is:

[2] Examples of this ISO combination are *en-gb* and *en-us*

https://pimtest.compano.com/API/jsonfeed/Products with related items for webshop/1/1?filter=code=man001**&culture=en-gb**

An example of the JSON output:

- The database language is *Dutch*
- The feed lay-out is multilingual
- The languages selected on the feed lay-out are *Dutch*, *German* and *English*
- Filter is **&culture=**en-GB

Field Type	Lay-out Field	JSON output

Text	Code	"Code": {
		"Value": "MAN001"
		}

Multilingual Text	Brand	"Brand": {
		"Value": "Manufacturer"
		}

Multilingual Text	Product description	"Description": {
		"Value": "Product 1"
		}

Multilingual Text	Product Class Name	"ClassName": {
		"Value": "Cable length meter"
		}

Percentage	Perc of features filled	"PercOfFeaturesFilled": {
		"Value": 50
		}

Multiple select	User defined field 'Application'	"Application": [
		{
		"Value": "Municipal water supply",
		"ValueDescription": "Municipal water supply"
		},
		{
		"Value": "Process technology",
		"ValueDescription": "Process technology"
		}

Range	Cable diameter	"EC000500_EF002356": {
		"FeatureCode": "EF002356",
		"Description": "Cable diameter",
		"Label": "Cable diameter",
		"Domain": "Range",
		"Value": {
		"Min": 10,
		"Max": 50
		},
		"Unit": "mm",
		"Postfix": "Millimetre"
		}

Enum (single choice)	Indication	"ProductFeaturesEDynamic": {
		"EC000500_EF000350": {
		"FeatureCode": "EF000350",
		"Description": "Indication",
		"Label": "Indication",
		"Domain": "Enum",
		"Value": "EV005572",
		"ValueDescription": "Digital"

Text	Item Number	"Code": {
		"Value": "MAN001"
		}

Multilingual Text	Description	"Description": {
		"Value": "Item 1"
		}

Multilingual Text	Commercial Description	"CommercialDescription": {
		"Value": "An English commercial description of the item."
		}

Decimal	Price	"GrossPriceInfoPrice": { "Value": 199

		}
Boolean	Has discount	"HasCondition": { "Value": false, "ValueDescription": "No" }
Image	Image	"Image": { "Value": "/Data/Environments/000001/Attachment/Bijlage/cable_length_meter.jpg" }

12.1.2 User Language

A monolingual customer can generate a monolingual feed for a system in the language of the user that runs the feed (the webservices user). This language is used for the value description of:

- Enum fields (single choice)
- Multiselect (multiple choice)
- Boolean fields (true/false)
- Multilingual text fields that are marked as multilingual

An example of the JSON output

- The database language is *Dutch*
- There are no languages selected on the feed lay-out
- The feed lay-out is monolingual (on the tab Features Multilingual is set to 'No')
- The webservices user language is *nl-NL*

Field Type	Lay-out Field	JSON output
Text	Code	"Code": { "Value": "MAN001" }
Multilingual Text	Brand	"Brand": { "Value": "Manufacturer" }
Multilingual Text	Product description	"Description": { "Value": "Product 1" }
Multilingual Text	Product Class Name	"ClassName": { "Value": "Kabellengtemeter" }
Percentage	Perc of features filled	"PercOfFeaturesFilled": { "Value": 50 }
Multiple select	User defined field 'Application'	"Application": [{ "Value": "Municipal water supply", "ValueDescription": "Gemeentelijke watervoorziening" }, { "Value": "Process technology", "ValueDescription": "Procestechnologie" }
Range	Cable diameter	"EC000500_EF002356": { "FeatureCode": "EF002356", "Description": "Kabeldiameter", "Label": "Kabeldiameter", "Domain": "Range", "Value": { "Min": 10, "Max": 50 }, "Unit": "mm", "Postfix": "Millimeter" }

Enum (single choice)	Indication	"ProductFeaturesEDynamic": {
		"EC000500_EF000350": {
		"FeatureCode": "EF000350",
		"Description": "Indicatie/aanduiding",
		"Label": "Indicatie/aanduiding",
		"Domain": "Enum",
		"Value": "EV005572",
		"ValueDescription": "Digitaal"

Text	Item Number	"Code": {
		"Value": "MAN001"
		}

Multilingual Text	Description	"Description": {
		"Value": "Artikel 1"
		}

Multilingual Text	Commercial Description	"CommercialDescription": {
		"Value": "Een Nederlandstalige commerciële omschrijving van het artikel."
		}

Decimal	Price	"GrossPriceInfoPrice": {
		"Value": 199
		}

Boolean	Has discount	"HasCondition": {
		"Value": false,
		"ValueDescription": "Nee"
		}

Image	Image	"Image": {
		"Value": "/Data/Environments/000001/Attachment/Bijlage/cable_length_meter.jpg"
		}

12.1.3 Lay-out Languages

If the customer has multilingual data, then the feed could contain multiple languages.

An example of the JSON output:

- The database language is *Dutch*.
- The languages selected on the feed lay-out are *Dutch*, *German* and *English*
- The feed lay-out is multilingual

Entity	Field Type	Lay-out Field	JSON output

Product	Text	Code	"Code": {
			"Value": "MAN001"
			}

Product	Multilingual Text	Brand (multilingual set to 'No')	"Brand": {
			"Value": "Manufacturer"
			}

Product	Multilingual Text	Product description	"Description": {
			"Value": {
			"nl-NL": "Product 1",
			"de-DE": "Produkt 1",
			"en-GB": "Product 1"
			}

Product	Multilingual Text	Product Class Name	"ClassName": {
			"Value": {
			"nl-NL": "Kabellengtemeter",
			"de-DE": "Kabellängenmessgerät",
			"en-GB": "Cable length meter"
			}

Product	Percentage	Perc of features filled	"PercOfFeaturesFilled": {
			"Value": 50
			}

Product	Multiple select	User defined field 'Application'	"Application": [
			{
			"Value": "Municipal water supply",
			"ValueDescription": {

			"nl-NL": "Gemeentelijke watervoorziening",
			"de-DE": "Munizipale Wasseranlieferung",
			"en-GB": "Municipal water supply"
			}
			},
			{
			"Value": "Process technology",
			"ValueDescription": {
			"nl-NL": "Procestechnologie",
			"de-DE": "Prozesstechnologie",
			"en-GB": "Process technology"
			}

Product feature	Range	Cable diameter	"EC000500_EF002356": {
			"FeatureCode": "EF002356",
			"Description": {
			"nl-NL": "Kabeldiameter",
			"de-DE": "Kabeldurchmesser",
			"en-GB": "Cable diameter"
			},
			"Label": {
			"nl-NL": "Kabeldiameter",
			"de-DE": "Kabeldurchmesser",
			"en-GB": "Cable diameter"
			},
			"Domain": "Range",
			"Value": {
			"Min": 10,
			"Max": 50
			},
			"Unit": "mm",
			"Postfix": {
			"nl-NL": "Millimeter",
			"de-DE": "Millimeter",
			"en-GB": "Millimetre"
			}

Product feature	Enum (single choice)	Indication	"ProductFeaturesEDynamic": {
			"EC000500_EF000350": {
			"FeatureCode": "EF000350",
			"Description": {
			"nl-NL": "Indicatie/aanduiding",
			"de-DE": "Anzeige",
			"en-GB": "Indication"
			},
			"Label": {
			"nl-NL": "Indicatie/aanduiding",
			"de-DE": "Anzeige",
			"en-GB": "Indication"
			},
			"Domain": "Enum",
			"Value": "EV005572",
			"ValueDescription": {
			"nl-NL": "Digitaal",
			"de-DE": "digital",
			"en-GB": "Digital"
			}

Item	Text	Item Number	"Code": {
			"Value": "MAN001"
			}

Item	Multilingual Text	Description	"Description": {
			"Value": {
			"nl-NL": "Artikel 1",
			"de-DE": "Artikel 1",
			"en-GB": "Item 1"
			}

Item	Multilingual Text	Commercial Description	"CommercialDescription": {
			"Value": {
			"nl-NL": "Een Nederlandstalige commerciële omschrijving van het artikel.",

			"de-DE": "Eine deutschsprachige kommerzielle Beschreibung des Artikels.", "en-GB": "An English commercial description of the item." }
Item	Decimal	Price	"GrossPriceInfoPrice": { "Value": 199 }
Item	Boolean	Has discount	"HasCondition": { "Value": false, "ValueDescription": { "nl-NL": "Nee", "de-DE": "Nein", "en-GB": "No" }
Item	Image	Image	"Image": { "Value": "/Data/Environments/000001/Attachment/Bijlage/cable_length_meter.jpg" }

12.1.4 Database Language

The database language can be set for the description of multilingual text fields that are configured as not multilingual at feed lay-out field level. An example is (product) Brand, it does not need to be translated.

13 Automatic image resizing

It is usually not necessary to add images of varying size. Images will be automatically resized when requested, using the parameters `W` and `H` in the URL:

```
http://name.compano.nl/Data/Environments/00XXXX/Images/ProductGroup/
Drawings/D1112.t.jpg?W=300&H=300
```

The example above will generate a resized (300x300) version of the JPG-file `D1112.t.jpg` and store it in a cache folder 300x300 on the Compano server.

Note: cache folders will be emptied when the original image is replaced. The new image will be resized on the next retrieval request.

The following image types can be resized this way:

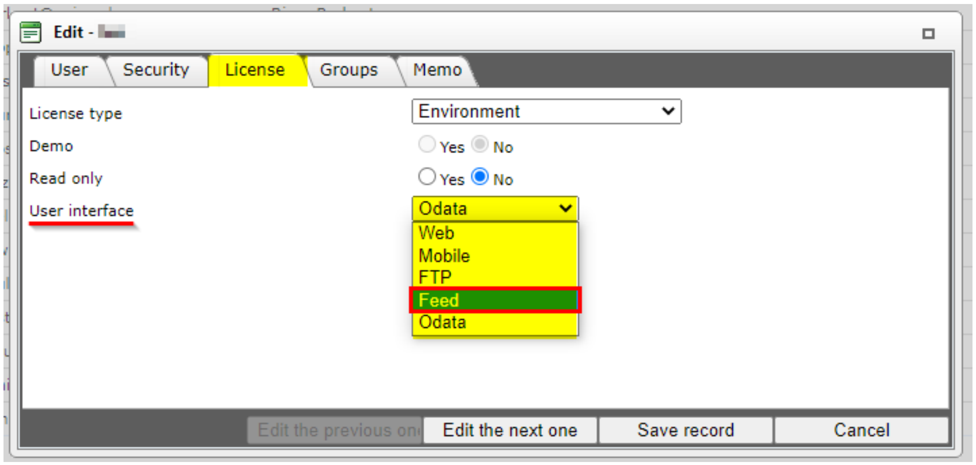
- PNG
- JPG
- JPEG
- GIF
- WMF

14 Appendix A: Changes to JSON feeds in L03

Compared to version L02, a number of changes have been made to the way JSON feeds work. A number of data fields that can be used in JSON feeds have also been renamed.

14.1 Feed-users

Existing 'users' which are used to access JSON- or XML-feeds need to be set to type *Feed*. User credentials for this type can no longer be used to login to the User Interface. Furthermore, the API-key associated with the 'feed user' account will only be visible to the Admin user of the system.


Edit -

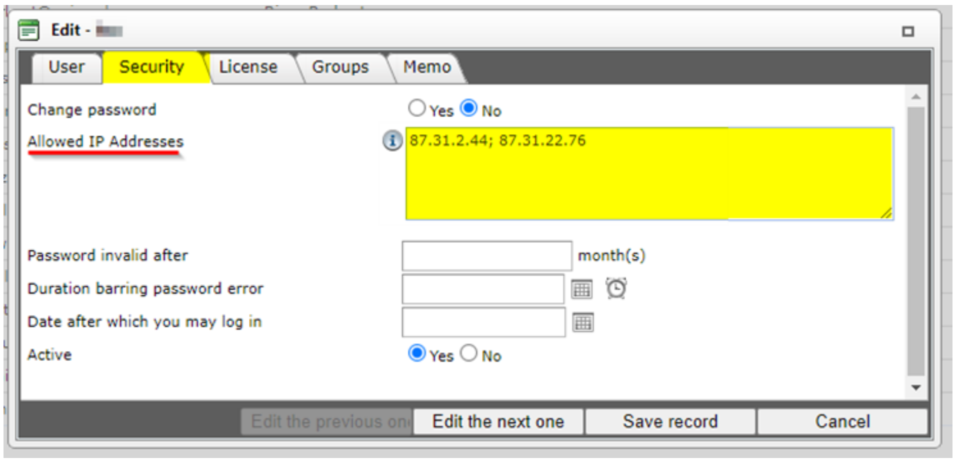
User | Security | License | Groups | Memo

License type: Environment
Demo: Yes [✓ No]
Read only: Yes [✓ No]
User interface: Odata
Web
Mobile
FTP
Feed
Odata

Edit the previous one | Edit the next one | Save record | Cancel

Note: Data access through the API-key can, optionally, be restricted based on IP-addresses:

- Multiple IP addresses are separated by a semicolon
- IP address ranges are allowed, for example: 86.77.22.0/24

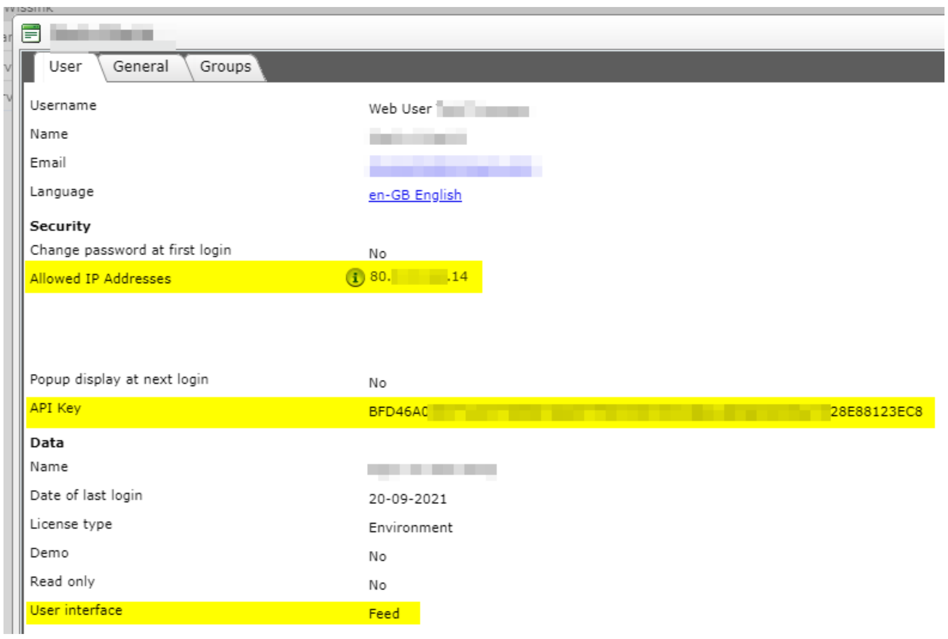

Edit -

User | Security | License | Groups | Memo

Change password: Yes [✓ No]
Allowed IP Addresses: 87.31.2.44; 87.31.22.76
Password invalid after: month(s)
Duration barring password error:
Date after which you may log in:
Active: [✓ Yes] No

Edit the previous one | Edit the next one | Save record | Cancel

Important: The API-key for the feed-user will only show after both steps mentioned above have been completed:



User | General | Groups

Username	Web User [redacted]
Name	[redacted]
Email	[redacted]
Language	en-GB English
Security	
Change password at first login	No
Allowed IP Addresses	80.[redacted].14
Popup display at next login	No
API Key	BFD46AC[redacted]28E88123EC8
Data	
Name	[redacted]
Date of last login	20-09-2021
License type	Environment
Demo	No
Read only	No
User interface	Feed

14.2 JSON feed sorting

Sorting of JSON feeds can now be set by adding fields on the sorting tab of the feed layout.

- Sorting can be set to either Ascending or Descending
- Sorting on multiple data fields is hierarchical

Note: Sorting is only valid for the main feed; any associated sub-feeds will not be sorted according to the settings of the main feed.

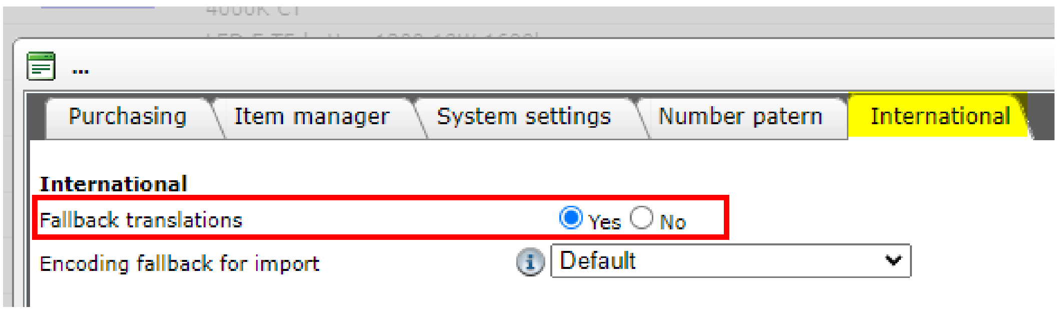
14.3 ModificationCode (MC/Mutation Code)

The ModificationCode (MC/Mutation Code) is no longer available in JSON feeds:

- Replacement field on **Product**: *Availability* (Deleted, Valid, Running in, Running out)
- Substitute field on **Attachment**: *IsArchived* (True / False)

14.4 Fallback of translation in feeds

In COS L03 the fallback of translation in feed is now dependent on the central fallback setting at *My Details > Compano Settings >* tab *International*.



...

Purchasing | Item manager | System settings | Number patern | International

International

Fallback translations	[✓ Yes] No
Encoding fallback for import	Default

When set to *Yes*, translatable fields in the feed will follow the standard fallback scheme; when set to *No*, translatable fields in the field will not fallback and thus remain empty.

Important: Please pay attention to this setting if your feeds (XML or JSON) contain multiple language fields.

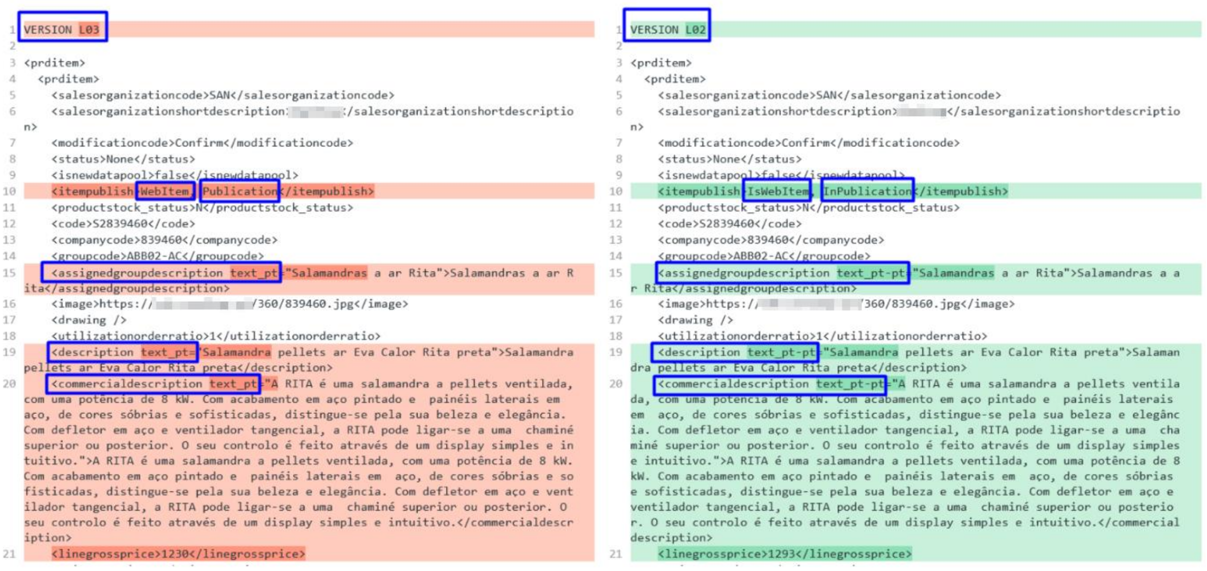
14.5 Renamed data fields

L02 field name	L03 field name
On screen Item Search	
Mutation code	ModificationCode
Available	Status code
#IsNewDataPool	Datapool
Group Code	Item Group code
Group (masterdata) description	Group (masterdata) description (item group)
Product manufacturer code	Product manufacturer code (gln)
Modification Time	Last modification
#IsWebItem	On internet
Minimum purchase in Order Units	Minimum purchase in OU
On screen Accessory Item	
#AddOnCode	Accessories item code
On screen Attachments	
File	Location
On screen Product Search	
Manufacturer code	Manufacturer code (gln)
Mutation code	ModificationCode
Description	Description (product)
Commercial description	Commercial description (product)
Brand	Brand (product)
Series	Series (product)
Type	Type (product)

14.6 Cancelled data fields

- Has accessory
- Choice item description

14.7 Structural and tag changes

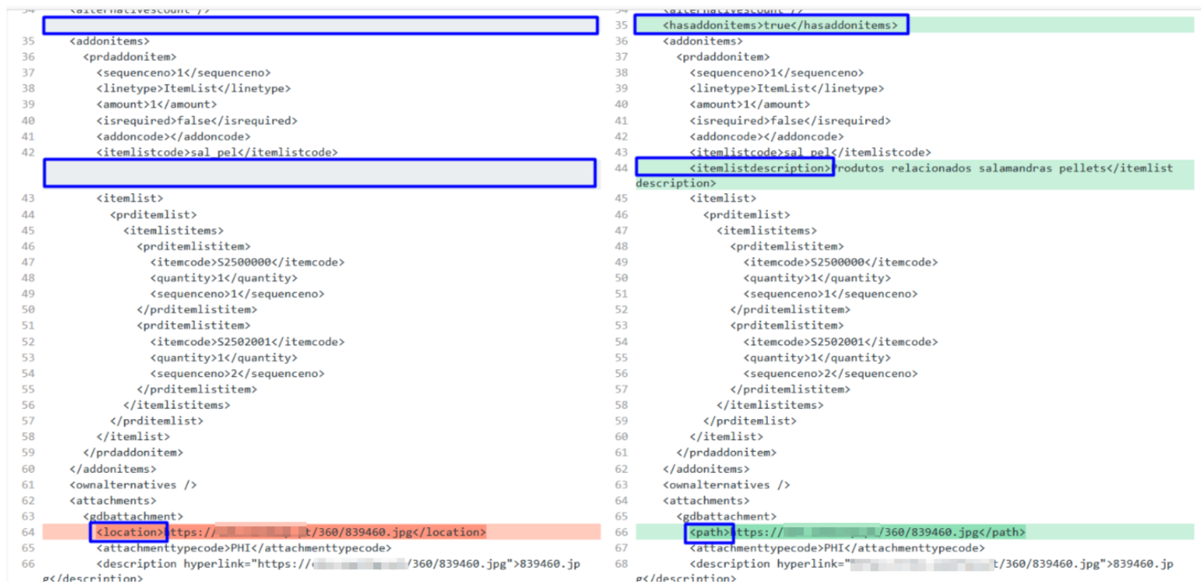


```
VERSION L03

<prditem>
  <prditem>
    <salesorganizationcode>SAN</salesorganizationcode>
    <salesorganizationshortdescription>[redacted]</salesorganizationshortdescription>
    <modificationcode>Confirm</modificationcode>
    <status>None</status>
    <isnewdatapool>false</isnewdatapool>
    <itempublish>WebItem, Publication</itempublish>
    <productstock_status>N</productstock_status>
    <code>S2839460</code>
    <companycode>839460</companycode>
    <groupcode>ABB02-AC</groupcode>
    <assignedgroupdescription text_pt="Salamandras a ar Rita">Salamandras a ar Rita</assignedgroupdescription>
    <image>https://[redacted]/360/839460.jpg</image>
    <drawing />
    <utilizationorderratio>1</utilizationorderratio>
    <description text_pt="Salamandra pellets ar Eva Calor Rita preta">Salamandra pellets ar Eva Calor Rita preta</description>
    <commercialdescription text_pt="A RITA é uma salamandra a pellets ventilada, com uma potência de 8 kW. Com acabamento em aço pintado e  painéis laterais em aço, de cores sóbrias e sofisticadas, distingue-se pela sua beleza e elegância. Com defletor em aço e ventilador tangencial, a RITA pode ligar-se a uma  chaminé superior ou posterior. O seu controlo é feito através de um display simples e intuitivo.">A RITA é uma salamandra a pellets ventilada, com uma potência de 8 kW. Com acabamento em aço pintado e  painéis laterais em  aço, de cores sóbrias e sofisticadas, distingue-se pela sua beleza e elegância. Com defletor em aço e ventilador tangencial, a RITA pode ligar-se a uma  chaminé superior ou posterior. O seu controlo é feito através de um display simples e intuitivo.</commercialdescription>
    <linegrossprice>1230</linegrossprice>
```

```
VERSION L02

<prditem>
  <prditem>
    <salesorganizationcode>SAN</salesorganizationcode>
    <salesorganizationshortdescription>[redacted]</salesorganizationshortdescription>
    <modificationcode>Confirm</modificationcode>
    <status>None</status>
    <isnewdatapool>false</isnewdatapool>
    <itempublish>IsWebItem, InPublication</itempublish>
    <productstock_status>N</productstock_status>
    <code>S2839460</code>
    <companycode>839460</companycode>
    <groupcode>ABB02-AC</groupcode>
    <assignedgroupdescription text_pt-pt="Salamandras a ar Rita">Salamandras a ar Rita</assignedgroupdescription>
    <image>https://[redacted]/360/839460.jpg</image>
    <drawing />
    <utilizationorderratio>1</utilizationorderratio>
    <description text_pt-pt="Salamandra pellets ar Eva Calor Rita preta">Salamandra pellets ar Eva Calor Rita preta</description>
    <commercialdescription text_pt-pt="A RITA é uma salamandra a pellets ventilada, com uma potência de 8 kW. Com acabamento em aço pintado e  painéis laterais em  aço, de cores sóbrias e sofisticadas, distingue-se pela sua beleza e elegância. Com defletor em aço e ventilador tangencial, a RITA pode ligar-se a uma  chaminé superior ou posterior. O seu controlo é feito através de um display simples e intuitivo.">A RITA é uma salamandra a pellets ventilada, com uma potência de 8 kW. Com acabamento em aço pintado e  painéis laterais em  aço, de cores sóbrias e sofisticadas, distingue-se pela sua beleza e elegância. Com defletor em aço e ventilador tangencial, a RITA pode ligar-se a uma  chaminé superior ou posterior. O seu controlo é feito através de um display simples e intuitivo.</commercialdescription>
    <linegrossprice>1293</linegrossprice>
```



```
    <alternativescount />

    <addonitems>
      <prdaddonitem>
        <sequenceno>1</sequenceno>
        <linetype>ItemList</linetype>
        <amount>1</amount>
        <isrequired>false</isrequired>
        <addoncode></addoncode>
        <itemlistcode>sal_pel</itemlistcode>

        <itemlist>
          <prditemlist>
            <itemlistitems>
              <prditemlistitem>
                <itemcode>S2500000</itemcode>
                <quantity>1</quantity>
                <sequenceno>1</sequenceno>
              </prditemlistitem>
              <prditemlistitem>
                <itemcode>S2502001</itemcode>
                <quantity>1</quantity>
                <sequenceno>2</sequenceno>
              </prditemlistitem>
            </itemlistitems>
          </prditemlist>
        </itemlist>
      </prdaddonitem>
    </addonitems>
    <ownalternatives />
    <attachments>
      <gdbattachment>
        <location>https://[redacted]t/360/839460.jpg</location>
        <attachmenttypecode>PHI</attachmenttypecode>
        <description hyperlink="https://[redacted]/360/839460.jpg">839460.jpg</description>
```

```
    <alternativescount />
    <hasaddonitems>true</hasaddonitems>
    <addonitems>
      <prdaddonitem>
        <sequenceno>1</sequenceno>
        <linetype>ItemList</linetype>
        <amount>1</amount>
        <isrequired>false</isrequired>
        <addoncode></addoncode>
        <itemlistcode>sal_pel</itemlistcode>
        <itemlistdescription>Produtos relacionados salamandras pellets</itemlistdescription>
        <itemlist>
          <prditemlist>
            <itemlistitems>
              <prditemlistitem>
                <itemcode>S2500000</itemcode>
                <quantity>1</quantity>
                <sequenceno>1</sequenceno>
              </prditemlistitem>
              <prditemlistitem>
                <itemcode>S2502001</itemcode>
                <quantity>1</quantity>
                <sequenceno>2</sequenceno>
              </prditemlistitem>
            </itemlistitems>
          </prditemlist>
        </itemlist>
      </prdaddonitem>
    </addonitems>
    <ownalternatives />
    <attachments>
      <gdbattachment>
        <path>https://[redacted]/360/839460.jpg</path>
        <attachmenttypecode>PHI</attachmenttypecode>
        <description hyperlink="[redacted]t/360/839460.jpg">839460.jpg</description>
```

15 Appendix B: Changes to JSON feeds in L04

Compared to version L03, a number of changes have been made to JSON feeds tags. This may lead to errors in Postman warning of obsolete fields.

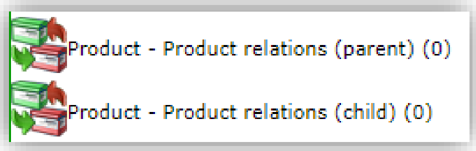
Note: As the product and item accessories system has been completely overhauled, in most cases best practice is to replace any faulty feed layouts with layouts that make use of the new product/item relations data fields.

15.1 Parts, Accessories, Alternatives and Predecessor/Successor

As of Compano software version L04, parts, accessories, alternatives and predecessor/successor have been replaced by ***product relations*** and ***item relations***. Consequently, for feeds containing Accessories and/or Predecessor/Successor codes, a number of changes have been made.

Relations can be set between a Product `(parent)` and Product `(child)` or Item `(parent)` and `Item (child)`.

Often, the relation can (or needs to) be set both ways. The Compano interface provides for both options, for example for Products:


Product - Product relations (parent) (0)
Product - Product relations (child) (0)

- **Product- Product relations (parent)**: Sets a relation from ***parent to child***, where the selected product (parent) is `product 1`, and the child product is `product 2`. In a JSON feed this is labelled as ***Relations1***.
- **Product- Product relations (child)**: Sets a relation from ***child to parent***, where the selected product (child) is `product 2`, and the parent product is `product 1`. In a JSON feed this is labelled as ***Relations2***.

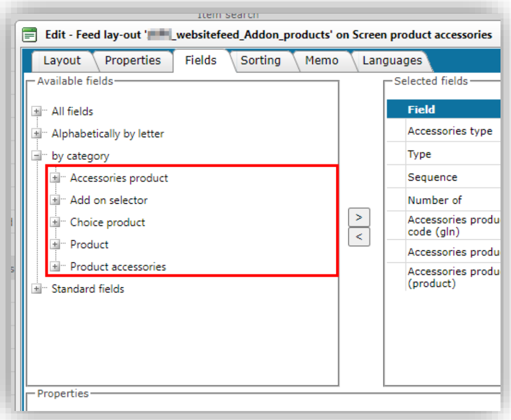
Note: Upon migration from L03 to L04, all product and item parts, accessories and alternatives are converted to ***Relations1***. Relations2 can be set as an ‘extra’, for instance to indicate that ‘this item/product can also be used with…’.

Important: Because of the changes to a number of tags, any feeds containing product and/or item relations need to be ***replaced*** !

15.1.1 Field selector

New categories have been added to the field selector to reflect the changes to product and item relations; example for Product relations:

L03 - Combined feeds for Product Accessories (including Parts:



Edit - Feed lay-out '[redacted]_websitefeed_Addon_products' on Screen product accessories

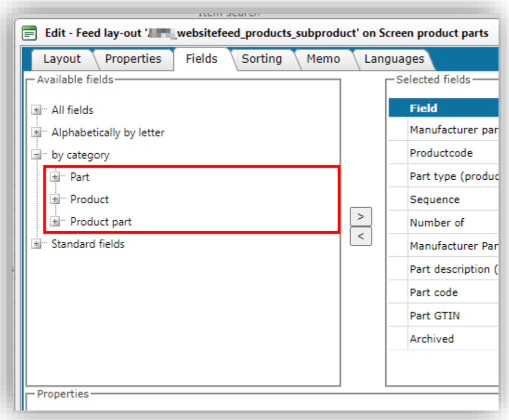
Layout | Properties | Fields | Sorting | Memo | Languages

Available fields
- All fields
- Alphabetically by letter
- by category
  - Accessories product
  - Add on selector
  - Choice product
  - Product
  - Product accessories
- Standard fields

Selected fields

Field
Accessories type
Type
Sequence
Number of
Accessories produ code (gln)
Accessories produ
Accessories produ (product)

Properties



Edit - Feed lay-out '[redacted]_websitefeed_products_subproduct' on Screen product parts

Layout | Properties | Fields | Sorting | Memo | Languages

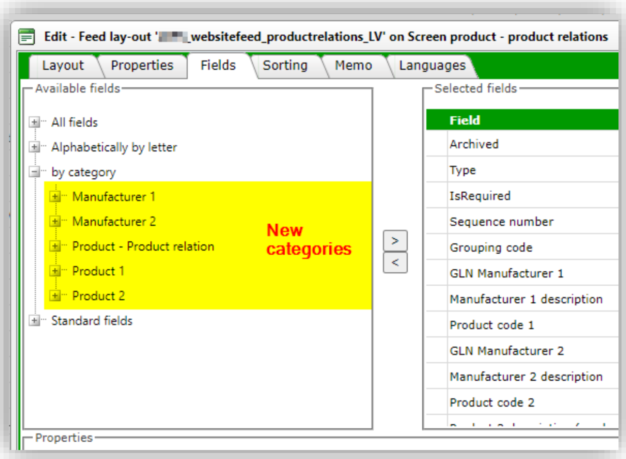
Available fields
- All fields
- Alphabetically by letter
- by category
  - Part
  - Product
  - Product part
- Standard fields

Selected fields

Field
Manufacturer par
Productcode
Part type (produc
Sequence
Number of
Manufacturer Par
Part description (
Part code
Part GTIN
Archived

Properties

L04 – One feed for Product relations (including Parts):



Edit - Feed lay-out '[redacted]_websitefeed_productrelations_LV' on Screen product - product relations

Layout | Properties | Fields | Sorting | Memo | Languages

Available fields
- All fields
- Alphabetically by letter
- by category
  - Manufacturer 1
  - Manufacturer 2
  - Product - Product relation
  - Product 1
  - Product 2
- Standard fields

Selected fields

Field
Archived
Type
IsRequired
Sequence number
Grouping code
GLN Manufacturer 1
Manufacturer 1 description
Product code 1
GLN Manufacturer 2
Manufacturer 2 description
Product code 2

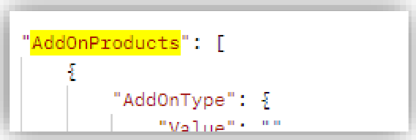
Properties

15.1.2 New data fields and feed tags

Feeds that have been setup for Parts, Accessories, Alternatives and/or Predecessor/Successor, need to be changed as a number of data fields have been redefined or have become obsolete.

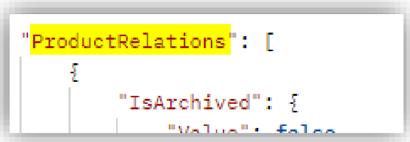
The main changes are:

- The tag `AddOnProducts` has been replaced by `ProductRelations`



```
"AddOnProducts": [
    {
      "AddOnType": {
        "Value": ""
```

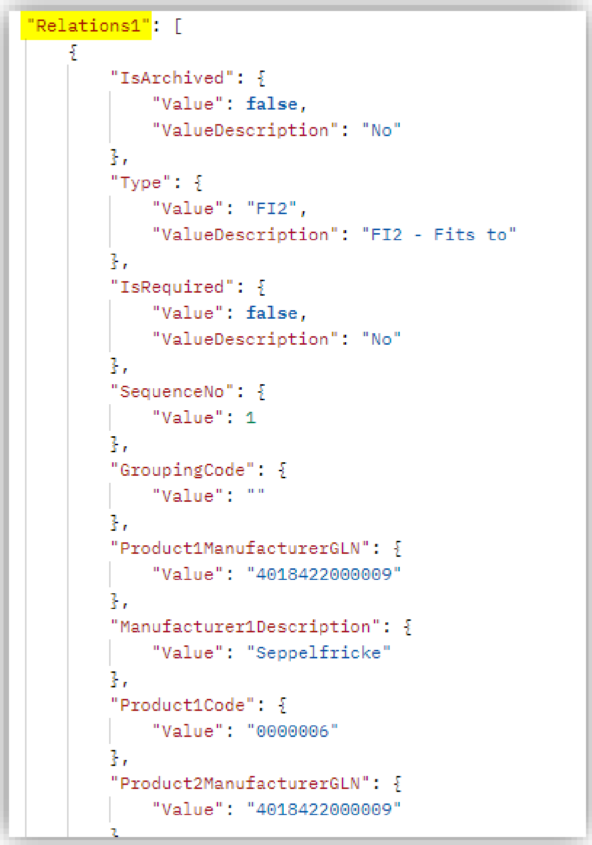
L03



```
"ProductRelations": [
    {
      "IsArchived": {
        "Value": false
```

L04

- The tag `AddOnItems` has been replaced by `ItemRelations`
- Parent relations are tagged by `Relations1` and child relations by `Relations2`



```
"Relations1": [
    {
        "IsArchived": {
            "Value": false,
            "ValueDescription": "No"
        },
        "Type": {
            "Value": "FI2",
            "ValueDescription": "FI2 - Fits to"
        },
        "IsRequired": {
            "Value": false,
            "ValueDescription": "No"
        },
        "SequenceNo": {
            "Value": 1
        },
        "GroupingCode": {
            "Value": ""
        },
        "Product1ManufacturerGLN": {
            "Value": "4018422000009"
        },
        "Manufacturer1Description": {
            "Value": "Seppelfricke"
        },
        "Product1Code": {
            "Value": "0000006"
        },
        "Product2ManufacturerGLN": {
            "Value": "4018422000009"
```

- Relations are always between `Product1` and `Product2`, or `Item1` and `Item2`, independent of whether `Product1` is the *parent* and `Product2` the *child* or vice versa; the direction of the relationship is determined by the tags `Relations1` and `Relations2`.
- Many other accessory-tags have been changed, for example:

Example: Changes to Product Accessories feed tags

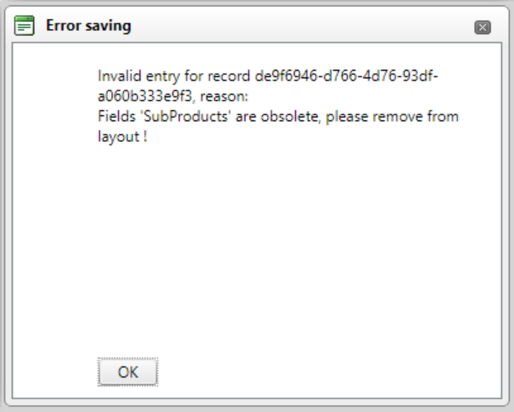
L03	L03 feed tag	L04	L04 feed tag
Manufacturer code (GLN)	`ManufacturerCode`	GLN Manufacturer 1	`Product1ManufacturerGLN`
Accessories product manufacturer code (gln)	`AddOnManufacturerCode`	GLN Manufacturer 2	`Product2ManufacturerGLN`
Manufacturer description	`ManufacturerDescription`	Manufacturer 1 description	`Manufacturer1Description`
Accessories product manufacturer description	`AddOnManufacturerDescription`	Manufacturer 2 Description	`Manufacturer2Description`
Product code	`ProductCode`	Product code 1	`Product1Code`
Accessories product code	`AddOnCode`	Product code 2	`Product2Code`

As many tags have been changed or have become obsolete, Compano recommends to replace any data feeds that include Product or Item Accessories or Alternatives.

15.1.3 Obsolete fields and feed tags

With the new Product and Item relations system, many data fields relating to Product and Item Accessories and Alternatives have become obsolete.

Tip: When checking feeds for obsolete fields, make sure to follow a bottom-up approach, as ***Postman*** (or other API platforms) will not indicate in which layout the obsolete field has been detected. Thus, always first check any sub-feed layouts for obsolete fields:

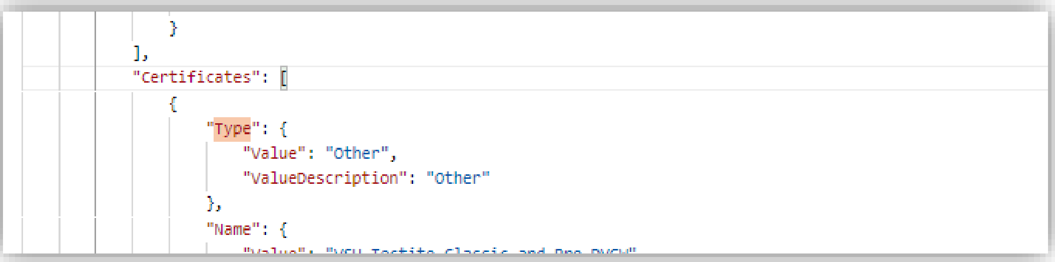


Error saving

Invalid entry for record de9f6946-d766-4d76-93df-a060b333e9f3, reason:
Fields 'SubProducts' are obsolete, please remove from layout !

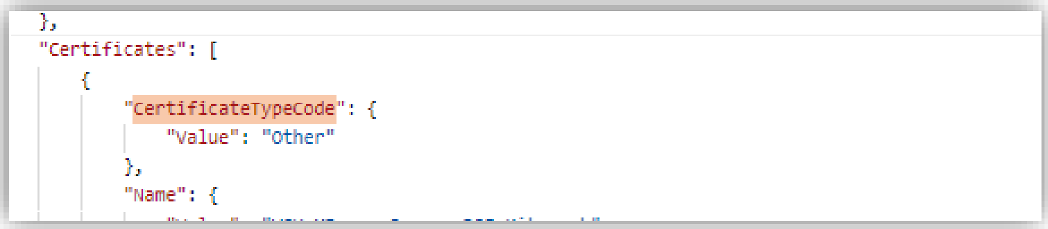
15.2 Certificates

The tag `Type` (certificate type) has been replaced by the tag `CertificateTypeCode`:



```
        }
    ],
    "Certificates": [
        {
            "Type": {
                "Value": "Other",
                "ValueDescription": "Other"
            },
            "Name": {
```

L03



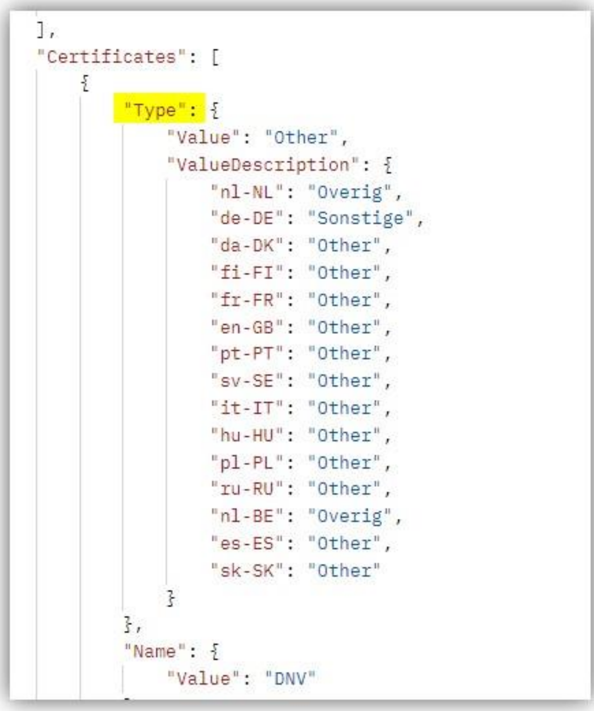
```
},
"Certificates": [
    {
        "CertificateTypeCode": {
            "Value": "Other"
        },
        "Name": {
```

L04

Note: This tag needs to be added to the a subfeed on the entity Products (!)

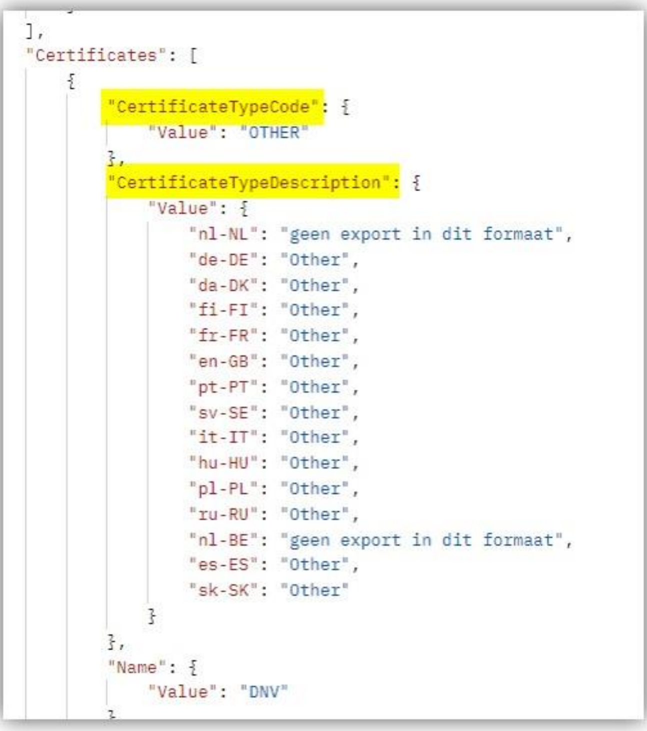
Important
The L03 tag `Type` was translatable. However, as of L04. this is no longer the case. Should you need to have translations of the Certificate type description, the `CertificateTypeDescription` field must be added.

L03: `Type`



```
],
"Certificates": [
    {
        "Type": {
            "Value": "Other",
            "ValueDescription": {
                "nl-NL": "Overig",
                "de-DE": "Sonstige",
                "da-DK": "Other",
                "fi-FI": "Other",
                "fr-FR": "Other",
                "en-GB": "Other",
                "pt-PT": "Other",
                "sv-SE": "Other",
                "it-IT": "Other",
                "hu-HU": "Other",
                "pl-PL": "Other",
                "ru-RU": "Other",
                "nl-BE": "Overig",
                "es-ES": "Other",
                "sk-SK": "Other"
            }
        },
        "Name": {
            "Value": "DNV"
```

L06: `CertificateTypeDescription`



```
],
"Certificates": [
    {
        "CertificateTypeCode": {
            "Value": "OTHER"
        },
        "CertificateTypeDescription": {
            "Value": {
                "nl-NL": "geen export in dit formaat",
                "de-DE": "Other",
                "da-DK": "Other",
                "fi-FI": "Other",
                "fr-FR": "Other",
                "en-GB": "Other",
                "pt-PT": "Other",
                "sv-SE": "Other",
                "it-IT": "Other",
                "hu-HU": "Other",
                "pl-PL": "Other",
                "ru-RU": "Other",
                "nl-BE": "geen export in dit formaat",
                "es-ES": "Other",
                "sk-SK": "Other"
            }
        },
        "Name": {
            "Value": "DNV"
```

15.1 Publication

When the data field *Publication* is empty, this will be handled as a null value:

L03: `“ProductPublish”: []`

L04: `“ProductPublish”: null`

The same is true for the JSON tag `ItemPublish`.