Tilted Channels – Products Export

Operator guide: designing a product data feed from a partner's template file

Version 1.1 · October 2026 · Applies to Tilted Channels for Sage 200 with the feed designer (Load Template)

1. What the Products export does

The Products export sends a file of stock item data from Sage 200 to a trading partner, marketplace or website. Each export is set up as a **Products module** in Tilted Channels. A module holds:

- **Which items** are included – active and/or inactive items, chosen product groups and analysis code filters.
- **The file format** – CSV, XML or JSON – plus the file name, compression and, for XML/JSON, the name of the node that holds each product.
- **The columns** – which Sage fields, fixed values and document-level header values are written, under what column names, in what order and with what formatting. This is the feed **design**.
- **Recipients** – one or more email addresses and/or FTP sites that receive the file.

Feeds are created on demand from the Product List form or automatically on a schedule, and are sent when they are submitted (posted). Creating a feed never sends anything on its own.

> **New:** the designer's **Load Template** button builds the column design from a sample file supplied by the partner. One column is created per header in the file, in the file's order. Columns whose names match Sage fields are mapped automatically; the rest are created as placeholders for you to map or fill in. This guide walks through that process end to end using a sample JSON template.

Where things are in Sage 200

Task	Menu location
Create or edit a Products module	Channels ▸ Channel List (Add / Edit)
Create feeds on demand	Channels ▸ Text & Excel Files ▸ Products ▸ Product List
View, submit and troubleshoot created feeds	Channels ▸ Export Feeds
FTP sites used by FTP recipients	Channels ▸ Settings ▸ FTP Sites tab
Scheduled creation and upload	Channels ▸ Settings ▸ Automation tab

2. The sample template used in this guide

A partner has asked for a JSON product feed in the layout below. Save this text as `product-template.json` – it is the file you will load into the designer. A template only needs one or two example products; the values themselves are ignored, only the field names and their positions matter.

```
{
  "feed": {
    "sender": "SAGE200",
    "generated": "2026-09-21T09:00:00",
    "version": "1.0"
  },
  "products": [
    {
      "Code": "ABC123",
      "Name": "Example product",
      "Barcode": "5012345678900",
```

```
      "CommodityCode": "33049900",
      "brand": "Example Brand",
      "category": "SKINCARE",
      "originCountry": "GB",
      "weightKg": "0.250",
      "price": "12.99",
      "currency": "GBP"
    },
    {
      "Code": "DEF456",
      "Name": "Second product",
      "Barcode": "5012345678917",
      "CommodityCode": "33049900",
      "brand": "Example Brand",
      "category": "HAIRCARE",
      "originCountry": "FR",
      "weightKg": "0.500",
      "price": "18.50",
      "currency": "GBP"
    }
  ]
}
```

Two things in this file matter for the setup:

- The products sit in an array called `products`. That name is the **Products Node** you enter in File Options before loading the template.
- The `feed` block sits outside the products array. Its three values are document-level **header** values: they are written once at the top of the file, not once per product.

> **Ask the partner for a template that contains every field they can accept**, even optional ones. Load Template creates columns only for the fields present in the file; a field missing from the template cannot be mapped later without adding it by hand.

3. Create the Products module

1. Open **Channels** ▸ **Channel List** and click **Add**.
2. Choose the **Products** channel. The wizard opens on the **Connection** page.
3. Enter a **Name** for the module (for example the partner's name) and tick **Enabled**. Tick **Include Sicon analysis code data** only if the site runs Sicon and the partner needs those fields.
4. Click **File Options** (see section 3.1), then **Filters** (see section 3.2). Click **Next >>**.
5. On the **Distribution** page add the recipients (section 3.3). Click **Next >>**.
6. On the **Email Details** page enter the subject and body used when the file is emailed. Click **Next >>**.
7. On the **Customisation** page open **Value Mapping** ▸ **Fields**. This opens the **Product Feed** designer, where you load the template (section 4).

3.1 File Options

File Options sets how the file is written. For the sample template:

Setting	Value for the sample	Notes
Format	JSON	CSV, XML, JSON or Excel. Excel cannot be used for product feeds.
Products Node	products	XML/JSON only. The element or array that holds each product. **Set this before loading the template** – it is how the designer tells product fields from header values.

Setting	Value for the sample	Notes
JSON array	ticked	Tick when the products node is a JSON array (`"products": [ ... ]`).
File starts with an array	clear	Tick only when the whole file is a bare array (`[ {...}, {...} ]`) with no surrounding object.
Default extension / Fixed filename	Default extension	Default extension names the sent file by feed number with `.json` (or `.csv` / `.xml`). Fixed filename sends the same name every time.
Compressed / Password	clear	Zip the file, optionally password protected.
Field Delimiter / Contains Headers	n/a	CSV only. For CSV templates the first row must contain the headers.

Click **Save** to close File Options.

3.2 Filters

Filters limits which stock items are exported. Leave everything at its default to export every item.

- **Stock Item Status** – tick Active and/or Inactive items.
- **Product Groups** – tick the groups to include (Check all / Invert help with long lists).
- **Warehouses** – tick one or more warehouses to export only the items held in them (an item needs a stock record in a ticked warehouse; it does not need stock). Leave every warehouse unticked to export items whatever warehouse they are in.
- **Analysis Codes** – click an analysis code in the top grid. If it has a list of values, tick the values to include in the bottom list; otherwise type the value to match in the Value column. An item must match every analysis code filter you set.

3.3 Distribution (recipients)

Each recipient receives its own copy of the file. Add at least one:

- **Add Email** – enter one or more email addresses. The file is attached to an email using the subject and body from the Email Details page.
- **Add FTP** – choose an FTP **Connection** (set up beforehand under Channels ▸ Settings ▸ FTP Sites) and an optional **Folder** on that site.
- **Add End Point** – post the file to a REST end point. Enter the **End point URL** (a full `http://` or `https://` address), the **REST method** (POST, PUT or PATCH) and, when the partner needs credentials, an **Authentication** (the `+` button creates one, `...` edits it: Basic, OAuth 2.0 or a bearer token, shared with the channel's other end points).

Use **Edit...** and **Remove** to change the list. Recipients can be ticked or unticked individually each time feeds are created.

An end point recipient also has an optional **Request body**. Leave it blank and the file itself is the body of the request. Fill it in when the partner's API wants the file wrapped inside a request of its own – for example a JSON-RPC call that takes the file as base64 text. The body is sent exactly as typed, with these tokens replaced:

- `[FILE_BASE64]` – the feed file, base64 encoded.
- `[FILE]` – the feed file as text, escaped for the body it sits in (JSON or XML).
- `[FILENAME]`, `[FEEDNUMBER]` – the file name and the export feed number.
- `[GUID]`, `[DATETIME]`, `[DATE]`, `[TIME]` – a new unique id and the time of sending.

A request body must use `[FILE_BASE64]` or `[FILE]`, otherwise the file would never be sent. A body that starts with `{` or `[` is posted as JSON and one that starts with `<` as XML. The request as sent is saved beside the feed as `<feed number>_Request.txt`. A reply of the form `{ "error": { ... } }` from a JSON-RPC API marks the feed **Processed With Errors** even though the HTTP status is 200.

> An end point recipient receives the file as the **body of the request**, with the content type of the file format (`application/json`, `application/xml` or `text/csv`) and the encoding from File Options. A compressed recipient's zip applies to email and FTP only. The reply is saved beside the feed as `<feed number>_Response.txt`; a non-2xx status, or a 2xx body that reports a rejection, marks the feed **Processed with errors** and records the reason.

4. Design the columns from the template

The **Product Feed** designer has two panels. **Available Data** (left) lists every field that can be exported: stock item fields, stock analysis codes, price bands and, when enabled, Sicon fields. **Included Data** (right) is the design – the columns that will be written, top to bottom in file order.

4.1 Load the template

1. Click **Load Template**. If the design already has columns you are asked to confirm, because the current design will be replaced (nothing is stored until you click Save). For XML and JSON a reminder appears that the template should contain all possible nodes – click OK.
2. Choose `product-template.json` and click Open.
3. The Included Data grid now shows one row per field in the file, in file order. Nothing has been saved yet.

This is what the sample template produces:

Type	Name	Column	Value	What happened
Header	doc.root.feed.sender	doc.root.feed.sender		Outside the products node → header value, written once. The column keeps its full path.
Header	doc.root.feed.generated	doc.root.feed.generated		Same.
Header	doc.root.feed.version	doc.root.feed.version		Same.
Item	Code	Code		Matched the Sage field **Code** automatically.
Item	Name	Name		Matched **Name**.
Item	Barcode	Barcode		Matched **Barcode**.
Item	Commodity Code	CommodityCode		Matched **Commodity Code** (spaces in the Sage name are ignored).
Constant	brand	brand		No Sage field of that name → empty placeholder to map or fill in.
Constant	category	category		Placeholder.
Constant	originCountry	originCountry		Placeholder.
Constant	weightKg	weightKg		Placeholder.
Constant	price	price		Placeholder.
Constant	currency	currency		Placeholder.

A header matches a Sage field when it equals the field's name in Available Data (with or without spaces, any case) or its internal field name, for example `Code`, `Name`, `Barcode`, `CommodityCode`, `PartNumber`, `ProductGroupCode`, `Weight`. Anything else becomes a Constant placeholder that keeps the partner's column name and position.

> Loading a JSON or XML template also records the products node's full path (here `doc.root.products`) in File Options so the export rebuilds the same structure. You do not need to change it.

4.2 Map the placeholders to Sage fields

A placeholder that should carry Sage data is **mapped**: it keeps its column name, position and format but takes its value from a field.

1. In **Included Data** click the placeholder row, for example `brand`.
2. In **Available Data** click exactly one field, for example `Manufacturer`.
3. Click **Map**. The row's Type changes to Item and its Name to the Sage field; the field disappears from Available Data (and, if the row was already mapped, its old field reappears there).

For the sample, map the placeholders like this:

Placeholder column	Map to (Available Data)	Type shown after mapping
brand	Manufacturer	Item
category	Product Group Code	Item
originCountry	Country of Origin Code	Item
weightKg	Weight	Item
price	Retail (inc VAT) – or whichever price band the partner should receive	Price Band

Any row can be mapped, including one that is already mapped. If a template header was auto-matched to the wrong Sage field, select that row, select the right field in Available Data and click **Map**: the row switches to the new field and the field it was using returns to Available Data.

4.3 Fill in the constants and header values

A placeholder that should always send the same text is a **constant**: type the text in its **Value** column. Header rows are filled in the same way. Tokens are replaced when the file is built:

Token	Sent as
[DATE]	the current date
[TIME]	the current time
[DATETIME]	the current date and time
[GUID]	a new unique identifier

For the sample:

Row	Value
currency	GBP
doc.root.feed.sender	SAGE200
doc.root.feed.generated	[DATETIME]
doc.root.feed.version	1.0

Use **Add Constant** or **Add Header** to add a column the template did not contain; type its name in the Column cell. Column names are limited to 64 characters and must be unique.

4.4 Formats

The **Format** column reshapes a value as it is sent. Leave it blank to send the value exactly as it is in Sage. Every format starts with `date:`, `num:` or `str:` – the full list is under the **Formatting & Constants** link on the designer. For the sample:

Row	Format	Effect
weightKg	num:0.000	0.404 rather than 0.40400
doc.root.feed.generated	date:yyyy-MM-dd HH:mm:ss	2026-09-21 19:36:51

Do not put a `num:` format on codes, SKUs or barcodes: it can strip leading zeros. Use `str:padleft(6,'0')` to zero-pad a code instead.

4.5 Save

Click **Save**. The whole design is checked first and every problem is reported together: a row with no column name, a column name over 64 characters, duplicated column names, or an invalid format. Fix them and Save again. **Cancel** discards everything since the designer was opened, including a loaded template.

The finished design for the sample looks like this:

Type	Name	Column	Value	Format
Header	doc.root.feed.sender	doc.root.feed.sender	SAGE200	
Header	doc.root.feed.generated	doc.root.feed.generated	[DATETIME]	date:yyyy-MM-dd HH:mm:ss
Header	doc.root.feed.version	doc.root.feed.version	1.0	
Item	Code	Code		
Item	Name	Name		
Item	Barcode	Barcode		
Item	Commodity Code	CommodityCode		
Item	Manufacturer	brand		
Item	Product Group Code	category		
Item	Country of Origin Code	originCountry		
Item	Weight	weightKg		num:0.000
Price Band	Retail (inc VAT)	price		
Constant	currency	currency	GBP	

Finish the wizard with **Next >>** and close it. The module is ready to create feeds.

5. What the exported file looks like

With the design above, the JSON feed produced from a test company begins like this (one product object per stock item, header values once at the top):

```
{
  "feed": { "sender": "SAGE200", "generated": "2026-09-21 19:36:51", "version": "1.0" },
  "products": [
    { "Code": "00035", "Name": "Bespoke Fragrance Customisation Replenishment 100ml",
      "Barcode": "886266000356", "CommodityCode": "33030010", "brand": "",
      "category": "BESP", "originCountry": "GB", "weightKg": "0.404",
      "price": "200.00000", "currency": "GBP" },
    { "Code": "00036", "Name": "Bespoke Fragrance Customisation 100ml",
      "Barcode": "886266000363", "CommodityCode": "49019900", "brand": "",
      "category": "BESP", "originCountry": "GB", "weightKg": "0.330",
      "price": "750.00000", "currency": "GBP" },
    ...
  ]
}
```

The columns appear in the order of the Included Data grid; dotted column names nest (`feed.sender` becomes `"feed": { "sender": ... }`). The file is written on one line; it is shown indented here for readability. `brand` is empty in this example because the test items have no Manufacturer.

6. Create and send feeds

6.1 On demand

1. Open **Channels** ▸ **Text & Excel Files** ▸ **Products** ▸ **Product List**.
2. In **Channels**, tick the Products module(s) to run. The module's recipients appear in **Recipients**, all ticked; untick any that should not receive this run.
3. Click **Create Full Feeds** to send every item, or **Create Delta Feeds** to send only the items that have changed since they were last sent (see section 6.3). The item data is read once per module and one feed is created per ticked recipient. A message confirms how many modules were processed.
4. Open **Channels** ▸ **Export Feeds** to see the feeds. **View** or **Payload** shows the file, **Open Folder** opens the folder it was written to, **Errors** shows any problem.
5. Tick the feeds and click **Submit** to send them – email recipients receive the attachment, FTP recipients receive the upload. Until a feed is submitted nothing leaves Sage.

6.2 Automatically

Under **Channels** ▸ **Settings** ▸ **Automation** the Products module offers two automations that run on the schedule you set: **Create product feeds** (a full feed), **Create delta product feeds** (changed items only) (creates a feed for every recipient of the module) and **Upload product feeds** (submits the feeds that are waiting). Schedule the upload after the create.

6.3 Full and delta feeds

A **full** feed contains every item that passes the filters. A **delta** feed contains only the items that are new or have changed since they were last sent, so a partner that keeps its own copy only receives what it needs to update.

- A item counts as changed when any value in a **designed column** is different from the value last sent. A change to a field that is not in the design does not count.
- A delta with nothing to send creates **no feed**. That is normal – it means nothing has changed.
- The first delta for a module, before any full feed, sends everything.
- A item that no longer passes the filters is simply left out. The feed does not tell the partner to delete it.
- Each feed is labelled in the Export Feeds list as `Full <date time>` or `Delta <date time>`.
- If a feed fails to send, **Submit** it again – it is the same file. If you **Roll Back** a feed instead, its items are treated as not sent and go out again on the next full or delta run.

> Run a full feed once when a partner is first set up, then schedule the delta. Changing the column design changes what is sent for every item, so the next delta after a design change is as large as a full feed.

7. CSV and XML templates

Load Template reads whichever format is selected in File Options, so set the format first.

- **CSV** – the first row must hold the column headers (tick Contains Headers and set the Field Delimiter to match). Each header becomes a column with that name. There are no header rows; an Add Header row in a CSV design is just an ordinary column.

- **XML** – enter the element that holds each product in **Products Node** (for example `Product` when the file is `<Catalogue><Product>...</Product></Catalogue>`). Elements below it become product columns; elements outside it become header values written once.
- **JSON** – as in this guide. When the file is a bare array with no surrounding object, tick **File starts with an array** and leave Products Node blank.

The same design works for any format: switching File Options from JSON to CSV writes the same columns as a flat file with the header row.

8. Troubleshooting

Symptom	Cause and fix
Every row came in as a Header after loading a JSON/XML template	Products Node was blank or did not match the array/element name in the file. Set it in File Options, then load the template again.
"'x' needs a column name" on Save	A row's Column cell is empty. Type the partner's column name.
"The columns below are duplicated ..."	Two rows share a column name. Rename or delete one.
"Column 'x' format '...' : ..."	The Format text is not a valid `date:` / `num:` / `str:` format. See Formatting & Constants.
Map is refused: "Select the column to map in the included data"	Click the row in Included Data first, then the field in Available Data, then Map.
Map is refused: "Select exactly one available field"	Select one row in Available Data before clicking Map.
A field is missing from Available Data	It is already used by a row in Included Data (each field can be used once), or – for analysis codes – the code has no friendly name in Sage, or Sicon data is not enabled on the Connection page.
The exported file is empty or missing items	Check Filters: status ticks, product groups and analysis code values must all match. Both status boxes unticked exports nothing.
Create Delta Feeds reports success but no feed appears	Nothing in the designed columns has changed since the items were last sent, so there is nothing to send. Use Create Full Feeds to send everything again.
A delta sends far more items than expected	The column design changed (every item now differs from what was sent), or the last feed was rolled back. Both are correct behaviour.
The template loaded but I clicked Cancel	Nothing was stored. Open the designer again and load the template.

Appendix A – Fields in Available Data

Stock item fields: ID, Code, Alias Code (the code this module's alias list gives the item – the code the partner knows it by – or the Sage code when the item has no alias), Name, Description, Barcode, Manufacturer, Part Number, Commodity Code, Country of Origin Code, Country of Origin Name, Weight, Tax Code, TaxRate, Unit, Product Group Code, Product Group Description, Stock Item Status, Inactive Date, Allow Sales Order, Default Pick Comments, Default Despatch Comments.

Plus: one entry per stock analysis code that has a name in Sage (the name shown is the analysis code's friendly name), one entry per price band (the band's name, sending the item's selling price in that band) and, when Include Sicon analysis code data is ticked, the Sicon analysis labels, Sicon distribution fields (pallet and box quantities, dimensions, stock cover, discontinued, ...) and Sicon warehouse fields.

Appendix B – Formatting reference

The same text is available from the **Formatting & Constants** link on the designer.

Dates and times – prefix date:

- `date:dd/MM/yyyy` → an incoming 2026-06-18 becomes 18/06/2026
- `date:yyyyMMdd` → 20260618
- `date:dd MMM yyyy` → 18 Jun 2026
- `date:dd/MM/yyyy|yyyy-MM-dd` → a UK date 18/06/2026 becomes 2026-06-18 (the part after `|` is the layout the value arrives in)
- `date:dd/MM/yyyy HH:mm` → includes the time

Layout letters: `dd` day, `MM` month, `yyyy` four-digit year, `yy` two-digit year, `HH` hour (24-hour), `hh` hour (12-hour), `mm` minutes, `ss` seconds, `tt` AM/PM. Case matters – capital `MM` is month, lower-case `mm` is minutes.

Numbers – prefix num:

- `num:#,##0.00` → 1234.5 becomes 1,234.50
- `num:0.00` → 1234.5 becomes 1234.50 (no thousands separator)
- `num:000000` → 625 becomes 000625

`0` forces a digit, `#` shows one only if present. Do not use `num:` on product codes, SKUs or barcodes.

Text – prefix str: (chain steps with |)

- `str:upper` / `str:lower` → change case
- `str:trim` → remove leading and trailing spaces
- `str:left(10)` / `str:right(4)` → keep the first / last characters
- `str:padleft(6,'0')` → 625 becomes 000625
- `str:replace('-','')` → 63124-1659 becomes 631241659
- `str:default('N/A')` → if the value is empty, send N/A instead
- `str:trim|upper` → trim, then upper-case