Tilted Channels – Customers Export

Operator guide: designing a customer 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 Customers export does

The Customers export sends a file of sales ledger customer data from Sage 200 to a trading partner, CRM, website or B2B platform. Each export is set up as a **Customers module** in Tilted Channels. A module holds:

- **Which accounts** are included – active and/or inactive accounts and analysis code filters.
- **How much of each account** – every contact, delivery address and price band, or the defaults only.
- **The file format** – CSV, XML or JSON – plus the file name, compression and, for XML/JSON, the name of the node that holds each customer.
- **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 Customer 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 that includes a contacts block.

Where things are in Sage 200

Task	Menu location
Create or edit a Customers module	Channels ▸ Channel List (Add / Edit)
Create feeds on demand	Channels ▸ Text & Excel Files ▸ Customer 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

How a customer is written

A customer account is one record. Fields that belong to the account itself (reference, name, address, terms, analysis codes) and constants sit at the record's root. Fields that come from a **contact**, a **delivery address** or a **price band** are written inside their own block under the customer, one entry per contact, address or band:

```
{ "Reference": "ABC001", "Name": "Example Customer Ltd", ...,
  "Contacts": { "Contact": [
      { "ContactName": "Jane Smith", "ContactEmail": "jane@example.com" }, ... ] },
  "DeliveryAddresses": { "DeliveryAddress": [ { ... } ] },
  "PriceBands": { "PriceBand": [ { ... } ] } }
```

In XML the same blocks are `<Contacts><Contact>...</Contact></Contacts>` and so on. In CSV every combination of contact, delivery address and price band produces a row, with the account fields repeated on each. Tick **Defaults only** on the Connection page to write just the account's preferred contact for its default role and its standard price band, which keeps one row or one contact entry per customer.

2. The sample template used in this guide

A partner has asked for a JSON customer feed in the layout below, with the contacts of each account nested inside it. Save this text as `customer-template.json` – it is the file you will load into the designer. A template only needs one or two example customers; the values are ignored, only the field names and their positions matter.

```
{
  "feed": {
    "sender": "SAGE200",
    "generated": "2026-09-21T09:00:00",
    "version": "1.0"
  },
  "customers": [
    {
      "Reference": "ABC001",
      "Name": "Example Customer Ltd",
      "Postcode": "SW1A 1AA",
      "CountryCode": "GB",
      "creditLimit": "5000.00",
      "paymentDays": "30",
      "currency": "GBP",
      "onHold": "false",
      "source": "SAGE200",
      "Contacts": {
        "Contact": [
          {
            "ContactName": "Jane Smith",
            "ContactEmail": "jane@example.com",
            "phone": "020 7946 0000",
            "role": "Accounts"
          }
        ]
      }
    },
    {
      "Reference": "DEF002",
      "Name": "Second Customer plc",
      "Postcode": "M1 1AA",
      "CountryCode": "GB",
      "creditLimit": "12000.00",
      "paymentDays": "60",
      "currency": "GBP",
      "onHold": "false",
      "source": "SAGE200",
      "Contacts": {
        "Contact": [
          {
            "ContactName": "John Jones",
            "ContactEmail": "john@example.com",
            "phone": "0161 496 0000",
            "role": "Buyer"
          }
        ]
      }
    }
  ]
}
```

Three things in this file matter for the setup:

- The customers sit in an array called `customers`. That name is the **Customers Node** you enter in File Options before loading the template.
- The `feed` block sits outside the customers array. Its three values are document-level **header** values: written once at the top of the file, not once per customer.
- The contacts are written as `"Contacts": { "Contact": [ ... ] }` – a block named after the Sage source (Contact, DeliveryAddress or PriceBand) with an `s`, holding an array of entries under the singular name. This is the shape the export produces, so a template in this shape loads straight back into the right block.

> **Ask the partner for a template that contains every field they can accept**, even optional ones, including at least one contact (and delivery address or price band if they take those). Load Template creates columns only for the fields present in the file.

3. Create the Customers module

1. Open **Channels** ▸ **Channel List** and click **Add**.
2. Choose the **Customers** 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 **Defaults only** if the partner wants one contact and one price band per account rather than all of them.
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 **Customer Feed** designer, where you load the template (section 4).

3.1 File Options

Setting	Value for the sample	Notes
Format	JSON	CSV, XML, JSON or Excel. Excel cannot be used for customer feeds.
Customers Node	customers	XML/JSON only. The element or array that holds each customer. **Set this before loading the template** – it is how the designer tells customer fields from header values.
JSON array	ticked	Tick when the customers node is a JSON array (`"customers": [ ... ]`).
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 accounts are exported. Leave everything at its default to export every account.

- **Customer Status** – tick Active and/or Inactive accounts. Both unticked exports nothing.
- **Analysis Codes** – click a customer 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 account 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. 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 **Customer Feed** designer has two panels. **Available Data** (left) lists every field that can be exported, grouped by type: Customer (account fields), Contact, Delivery Address, Price Band and Analysis Code. **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 `customer-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	What happened
Header	doc.root.feed.sender	doc.root.feed.sender	Outside the customers 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.
Customer	Reference	Reference	Matched the Sage field **Reference** (the account number).
Customer	Name	Name	Matched **Name**.
Customer	Postcode	Postcode	Matched **Postcode**.
Customer	Country Code	CountryCode	Matched **Country Code** (spaces in the Sage name are ignored).
Customer	Credit Limit	creditLimit	Matched **Credit Limit** (case is ignored too).
Constant	paymentDays	paymentDays	No Sage field of that name → empty placeholder to map or fill in.
Constant	currency	currency	Placeholder.
Customer	On Hold	onHold	Matched **On Hold**.
Constant	source	source	Placeholder.
Contact	Contact Name	ContactName	Matched **Contact Name**; the `Contacts.Contact.` prefix is dropped because the export writes it inside the Contacts block.
Contact	Contact Email	ContactEmail	Matched **Contact Email**.
Constant	Contacts.Contact.phone	Contacts.Contact.phone	Placeholder. It keeps its full path until it is mapped.
Constant	Contacts.Contact.role	Contacts.Contact.role	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 `Reference`, `Name`, `Postcode`, `CountryCode`, `CreditLimit`, `ContactName`, `DeliveryPostcode`, `PriceBandName`. Anything else becomes a Constant placeholder that keeps the partner's column name and position.

> Loading a JSON or XML template also records the customers node's full path (here `doc.root.customers`) 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 `paymentDays`.
2. In **Available Data** click exactly one field, for example `Payment Terms Days`.
3. Click **Map**. The row's Type changes to the field's type 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).

When a placeholder inside a block (`Contacts.Contact.phone`) is mapped to a field of that block's type (a Contact field), the block prefix is dropped and the column becomes just `phone`, written inside each contact entry. For the sample, map the placeholders like this:

Placeholder column	Map to (Available Data)	Column and type after mapping
paymentDays	Payment Terms Days	paymentDays – Customer
currency	Currency Code	currency – Customer
Contacts.Contact.phone	Contact Telephone	phone – Contact
Contacts.Contact.role	Contact Role	role – Contact

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: `[DATE]`, `[TIME]`, `[DATETIME]` and `[GUID]`. For the sample:

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

A constant always sits at the customer's root, even when its template path was inside a block. 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 and in Appendix B. For the sample:

Row	Format	Effect
creditLimit	num:0.00	5000 rather than 5000.00000
doc.root.feed.generated	date:yyyy-MM-dd HH:mm:ss	2026-09-21 19:30:00

Do not put a `num:` format on account references or telephone numbers: it can strip leading zeros. Use `str:padleft(6,'0')` to zero-pad 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	
Customer	Reference	Reference		
Customer	Name	Name		
Customer	Postcode	Postcode		

Type	Name	Column	Value	Format
Customer	Country Code	CountryCode		
Customer	Credit Limit	creditLimit		num:0.00
Customer	Payment Terms Days	paymentDays		
Customer	Currency Code	currency		
Customer	On Hold	onHold		
Constant	source	source	SAGE200	
Contact	Contact Name	ContactName		
Contact	Contact Email	ContactEmail		
Contact	Contact Telephone	phone		
Contact	Contact Role	role		

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 begins like this: one object per account with its contacts nested, header values once at the top. The structure is exactly what a test run produced; the account details shown are illustrative.

```
{
  "feed": { "sender": "SAGE200", "generated": "2026-09-21 20:12:45", "version": "1.0" },
  "customers": [
    { "Reference": "EX0003", "Name": "Example Trading GmbH",
      "Postcode": "52062", "CountryCode": "DE", "creditLimit": "0.00", "paymentDays": "30",
      "currency": "GBP", "onHold": "False", "source": "SAGE200",
      "Contacts": { "Contact": [
        { "ContactName": "Anna Beispiel", "ContactEmail": "anna@example.de",
          "phone": "+49 241 000000", "role": "Account" },
        { "ContactName": "Anna Beispiel", "ContactEmail": "anna@example.de",
          "phone": "+49 241 000000", "role": "SendStatementTo" } ] } },
    { "Reference": "EX0149", "Name": "Sample Decor d.o.o.",
      "Postcode": "1000", "CountryCode": "SI", "creditLimit": "0.00", "paymentDays": "30",
      "currency": "EUR", "onHold": "False", "source": "SAGE200",
      "Contacts": { "Contact": [
        { "ContactName": "J Novak", "ContactEmail": "info@example.si",
          "phone": "00386 41 000000", "role": "Account" } ] } },
    ...
  ]
}
```

The columns appear in the order of the Included Data grid; Contact columns are collected into the `Contacts` block, one entry per contact and role (the same person can appear under several roles, as above). Tick **Defaults only** on the Connection page to write just the preferred contact of the default role. The file is written on one line; it is shown indented here for readability.

6. Create and send feeds

6.1 On demand

1. Open **Channels** ▸ **Text & Excel Files** ▸ **Customer List**.
2. In **Channels**, tick the Customers 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 customer, or **Create Delta Feeds** to send only the customers that have changed since they were last sent (see section 6.3). The customer data is read once per

module and one feed is created per ticked recipient. A message confirms how many modules were processed. A full customer file can be large – a company with tens of thousands of accounts produces a file of several megabytes.

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 Customers module offers two automations that run on the schedule you set: **Create customer feeds** (a full feed), **Create delta customer feeds** (changed customers only) and **Upload customer feeds**. Schedule the upload after the create.

6.3 Full and delta feeds

A **full** feed contains every customer that passes the filters. A **delta** feed contains only the customers 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 customer 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 customer 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 customers 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 customer, 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. Contact, delivery address and price band columns sit on the same row as the account; there are no blocks.
- **XML** – enter the element that holds each customer in **Customers Node** (for example `Customer` when the file is `<Accounts><Customer>...</Customer></Accounts>`). Elements below it become customer columns; a `<Contacts><Contact>` block inside it becomes Contact 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 Customers Node blank.

8. Troubleshooting

Symptom	Cause and fix
Every row came in as a Header after loading a JSON/XML template	Customers Node was blank or did not match the array/element name in the file. Set it in File Options, then load the template again.
A contact field went out at the customer root, or nested under an extra Contacts.Contact	The column was left as a constant, or was mapped to a Customer field instead of a Contact field. Map it to the Contact field; the block prefix is then dropped automatically.
"'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.
"Customer data is empty" or accounts are missing	Check Filters: both status boxes unticked exports nothing, and every analysis code filter must match.
Each account appears several times in a CSV	Expected when the account has several contacts, delivery addresses or price bands. Tick Defaults only on the Connection page for one row per account.
The template loaded but I clicked Cancel	Nothing was stored. Open the designer again and load the template.

Appendix A – Fields in Available Data

Customer: ID, Reference, Name, Short Name, Account Balance, Credit Limit, Currency Name, Currency Symbol, Currency Code, Country Code, Country Name, Use Tax Code As Default, Tax Code, Tax Rate, Tax Rate Name, Tax Registration Number, Default Order Priority, Default Nominal Account Number, Default Nominal Cost Centre, Default Nominal Department, Early Settlement Discount Percent, Day Early Settlement Discount Applies, Payment Terms Days, Payment Terms Name, Invoice Line Discount Percent, Invoice Discount Percent, On Hold, Current Order Value, Date Last Changed, Last Transaction Date, Trading Terms, Credit Reference, Average Time To Pay, Account Opened, Terms Agreed, Account Status, Status Reason, Address Line 1–4, City, County, Postcode, Country, Telephone Country Code, Telephone Area Code, Telephone Subscriber Number, Fax Country Code, Fax Area Code, Fax Subscriber Number, Website.

Contact: Contact Name, Contact Default Role, Contact Telephone, Contact Fax, Contact Email, Contact Website, Contact Mobile, Contact Preferred for Role, Contact Role.

Delivery Address: Delivery Description, Delivery Postal Name, Delivery Address Line 1–4, Delivery City, Delivery County, Delivery Postcode, Delivery Telephone, Delivery Fax, Delivery Email, Delivery Contact, Delivery Tax Registration Number, Delivery Address Default.

Price Band: Price Band Name, Price Band Type.

Analysis Code: one entry per customer analysis code that has a name in Sage (shown under its friendly name).

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 references or telephone numbers.

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