BarTender

BarTender Track & Trace™

Guide to barcode/RFID label design, printing/encoding, and pushing the identity up to the Track & Trace platform

Supports BarTender 12 and later versions

Document revision: 2.0

bartendersoftware.com

May 2026

The purpose of this guide is to help customers and partners design a BarTender document that prints and encodes an RFID label that's suitable for use with BarTender Track & Trace. Once the label is printed/encoded the EPC identity will be pushed up to the platform in the Zone of the user's choice at a user defined Location.

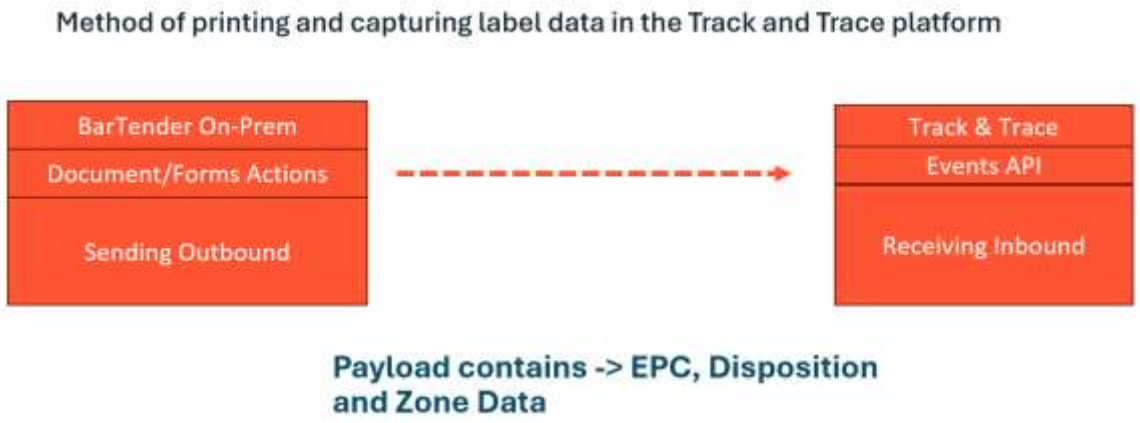
As we shall see, the implementation of pushing the identity to BarTender Track & Trace is achieved through the use of Document Actions, because as yet, there is no native connector between the two products.

This documentation assumes that the user already has good familiarity and experience with the **BarTender Designer** application, for label design and printing. The tutorial below will require the **Automation** edition of **version 12.0** or higher.

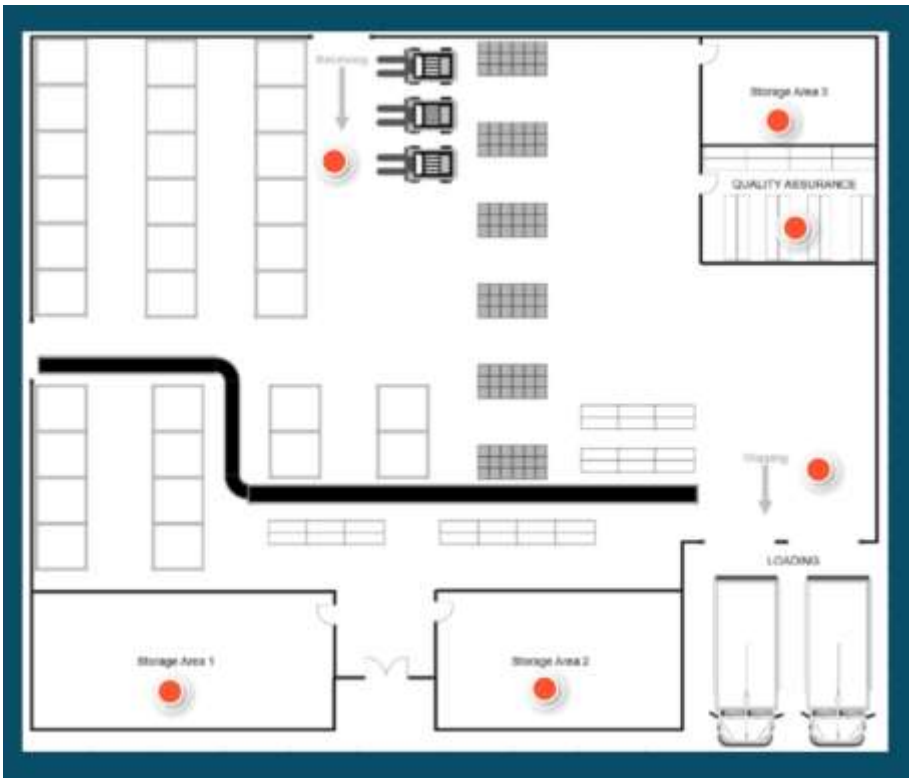
> An example BTW file, BarTender Track & Trace Sample Label, has been created and is available alongside this documentation.

Step by step workflow between BarTender Designer and BarTender Track & Trace

In the below diagram we present the basic idea of how, through implementation of BarTender Actions, the item's identity can be pushed to the Track & Trace platform as part of an automated process while the label is printed.



The printed identity will then be visible at the designated Zone within the map of your chosen Location when viewed in the Web Console of your BarTender Track & Trace tenant.



The Action to push the identity leverages the web API of the BarTender Track & Trace platform.

The BarTender Track & Trace platform has a fully-fledged and evolving set of APIs that allow for a great deal of capability to be accessed by other software and systems; this includes the BarTender Label software.

Regarding the use of the API

1. For the API calls in the below examples, the "{{root}}" tag indicates the base URL of the BarTender Track & Trace tenant in question. In my example this is:

"https://seagulldemo.bartender-tt.com". For yourselves, replace this base URL with the URL of the tenant that you are using.

2. The API Key should always be passed in any request using an HTTP header and is the approach used as the authentication method when making REST API calls to the BarTender Track & Trace platform:

   `Apikey: XXXXXXXXXXXXXXX`

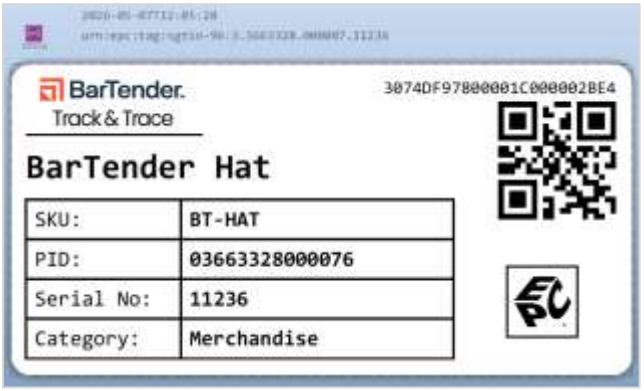
   Where XXXXXXXXXXXXXXX is replaced with the API Key extracted from the Web Console via the user profile.

3. For making API calls to BarTender Track & Trace for testing purposes, a REST Client such as **Postman** (https://www.postman.com) or **Insomnia** (https://insomnia.rest) can be used, or a **CURL** command from a command prompt window. An example of the CURL syntax is as below:

```
curl -X POST "https://{{root}}/statemachine-epcis-events-input-rest/rest/events"
-H "Content-Type: application/json"
-H "ApiKey: ABCDE12345"
-d '{"events": [
            {
               ---Event Data---
            }
      ]
    }'
```

From the BarTender Designer app

The goal is to design a BarTender document that contains each of the data elements in text, barcode, and RFID form. All of which is sufficient to properly identify a physical item, and then to track it through the business using the BarTender Track & Trace product.



BarTender Track & Trace Sample Label.btw

The document would also incorporate Actions that execute upon printing of each label. These Actions push the identity into the BarTender Track & Trace platform, delivering it to a desired Location and Zone, based on where the printing and applying of the label is taking place.

To assist with this, Seagull Software has provided an example template and made it available alongside this documentation. This template is located in the Template Browser/Template Manager and is accessible from there. When creating a new document with the New Document Wizard, choose the option Select Template from Library. Search for "BarTender Track & Trace Sample Label" to locate the template.

From the BarTender Track & Trace web console

User Credentials

To begin, sign into the BarTender Track & Trace Web Console with your user credentials, making sure the user has a role of Manager at the least.

Get an API Key for your signed in user:

1. Click the user profile icon in the top-right of the Home page.
2. Select **User Settings** and the **API Key** item in the list.

3. Click **View** and then **Copy**.

Note Keep the API Key value that you extract somewhere safe, so that it can be used when making any API calls to the BarTender Track & Trace platform.

Master Data

1. From the left-hand side bar click (Settings), then (Track & Trace), then along the top (Master Data).
2. Download or transcribe each of the products into a client-side *.csv file, or use the API call to do the equivalent:

   GET {{root}}/product-api/rest/products

   The API call returns JSON data in the response, from which you can parse fields such as:

   - gtin
   - productCode
   - productLabelShort
   - productLabelLong
   - categoryLevel1Code
   - companyPrefixLength

   You should have or assemble a *.csv or database file with a structure similar to the below example.

```
ProductLabelShort,ProductLabelLong,GTIN,ProductCode,CategoryLevel1Code,Image
Hat,BarTender Hat,03663328000076,BT-HAT,Merchandise,Picture1.png
Pen,BarTender Pen,03663328000083,BT-PEN,Merchandise,Picture2.png
Bottle Opener,BarTender Bottle Opener,03663328000090,BT-BTL,Merchandise,Picture3.png
Charging Cable,BarTender Charging Cable,03663328000106,BT-
CHG,Merchandise,Picture4.png
Long Sleeve Polo,BarTender Long Sleeve Polo,03663328000113,BT-
LSP,Merchandise,Picture5.png
Fleece Jacket,Seagull Fleece Jacket,03663328000120,SG-FJ,Merchandise,Picture6.png
T-Shirt,Seagull T-Shirt,03663328000137,SG-TS,Merchandise,Picture7.png
```

```
Short Sleeve Polo,Seagull Short Sleeve Polo,03663328000144,SG-
SSP,Merchandise,Picture8.png
```

Note As per the Reference Document label design, key fields to use would be: GTIN, ProductCode, ProductLabelLong, and CategoryLevel1Code. However, feel free to use whichever fields you deem to be most appropriate.

3. Return to the Home Page, and from the Location dropdown list on the left-hand side, select the location (the physical site: factory/warehouse/office/store) into which you want your items to be placed. Make a note of its name.

   Alternatively, use the API call:

   ```
   GET {{root}}/statemachine-api-
   configuration/rest/configuration/locations?level=premise
   ```

   The API call returns JSON data in the response, from which you can parse fields such as:

   - id
   - code
   - name
   - level
   - type

With the desired Location selected, you can make a note of all the names for each Zone set on the floor-plan map with a hot-spot icon.

However, in order to deliver an item to a specific zone when making an API call from the client-side, you need to know the ID that corresponds to the zone name.

The API call to do this is:

```
GET {{root}}/statemachine-api-
configuration/rest/configuration/fixture/premise/<code>
```

Where <code> is the location code from the previous call that returns a list of Locations that are established in your tenant. For example: LOCATION3

The API call returns JSON data in the response, from which you can parse fields such as:

- id
- name

Note As per the Location identification returned from the Track & Trace platform, it is the prefix to the name of the Location as it appears in the UI of the Home page, that corresponds to the code returned from the API call that is of interest. Make a careful note of all these details.

The list of ID's and names will be used in the BarTender document's Action that pushes the identity up to the Track & Trace platform. This could either be hardcoded into the action or be made selectable at print time using a list form control to set a Named Data Source.

Here is an example of what this data might look like:

```
id - SEAGULLDEMO.00004.1000000000002
name - Receiving
```

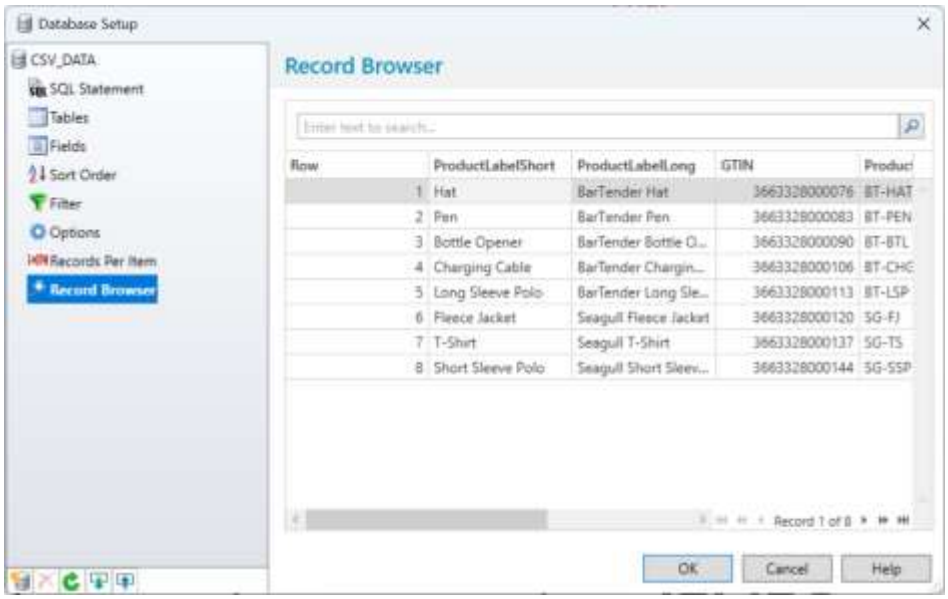
From the BarTender Designer app

Create a Document

1. In the BarTender Designer application create a **new document** and set the Page Setup to equal the RFID label media you're using.
2. Select the RFID capable Seagull printer driver that points to your RFID capable printer.

Note	At time of writing, the BarTender Track & Trace mobile app only supports UHF SGTIN-96 passive tags.

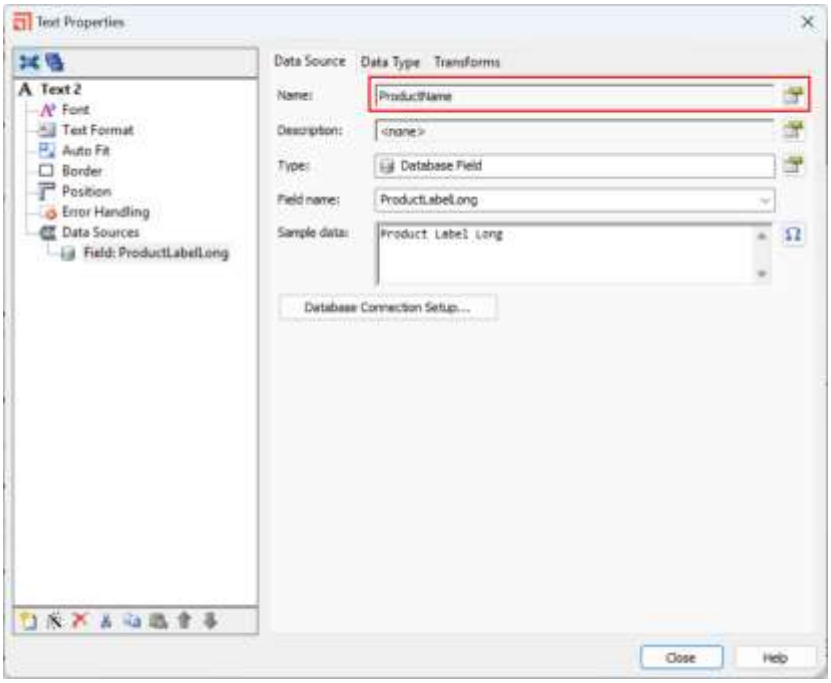
3. Make a database connection to the label using the data file you previously downloaded and assembled, or a database of type in which you store this Product data.
4. Step through the wizard to finalize the connection and confirm in the Record Browser that the product records are properly displayed.
5. Click **OK** when done.



Add Text Objects

- Using the database fields now visible in the Toolbox, drag and drop them as text objects into the label design to begin the process of designing the label.

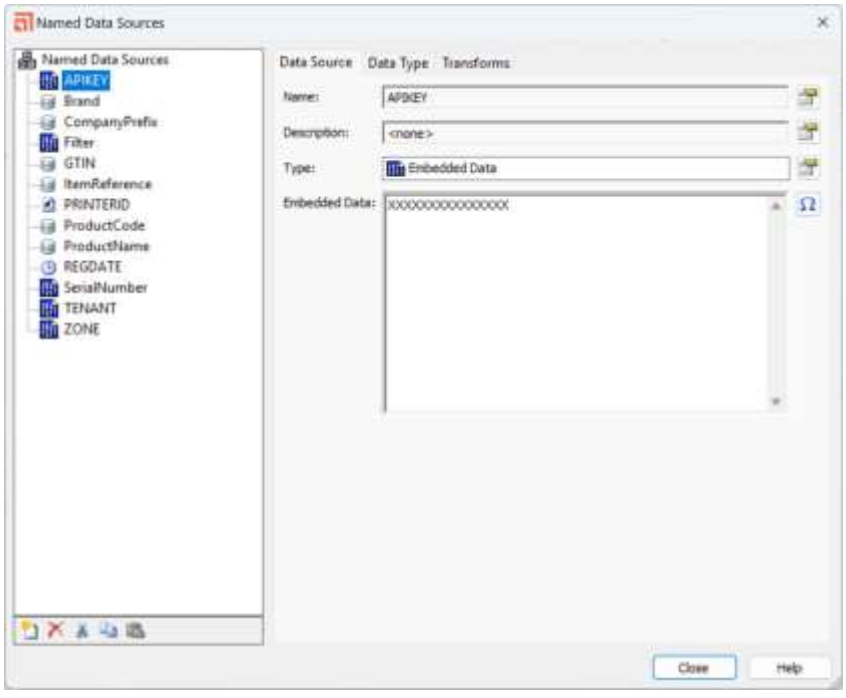
For each of the Database Field data sources, you can set a Name to make it a Named Data Source, and thus able to be referenced from within a BarTender Action that will be used to push data up to the Track & Trace platform.



Create Other Named Data Sources

Other Named Data Sources of various types should also be created for the purpose of BarTender Actions that execute when printing.

1. Right click the Named Data Sources icon in the ToolBox.
2. Select the Manage Named Data Sources item.

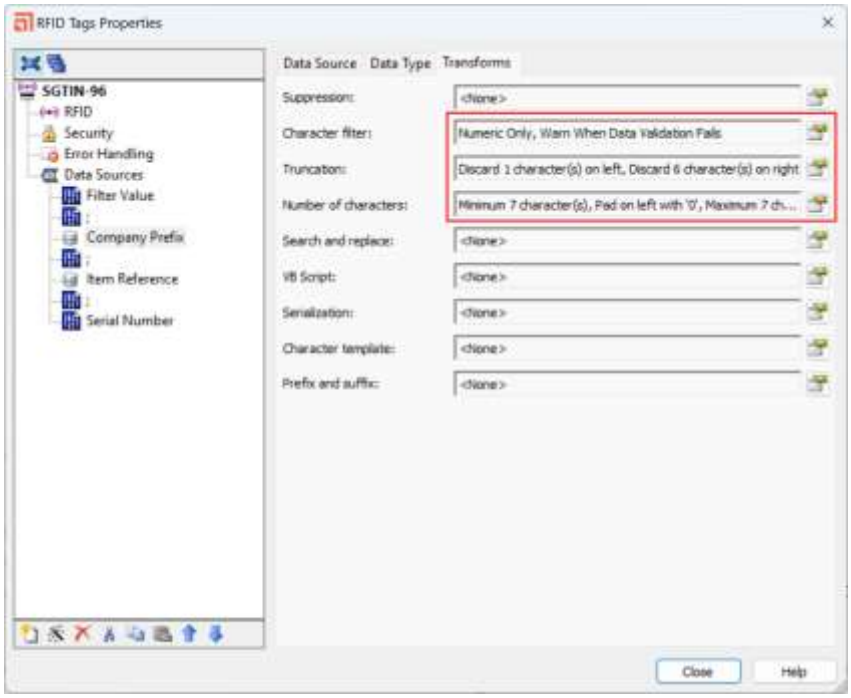


3. Create additional Named Data Sources as follows with names and data-source types as below:
    - **APIKEY** – Embedded Data
    - **Brand** – Database Field
    - **CompanyPrefix** – Database Field
    - **Filter** – Embedded Data
    - **GTIN** - Database Field
    - **ItemReference** – Database Field
    - **PRINTERID** – Print Control Field (Printer Name)
    - **ProductCode** – Database Field
    - **ProductName** – Database Field
    - **REGDATE** – Clock (custom pattern of: yyyy'-'MM'-'dd'T'HH:MM:ss)
    - **SerialNumber** – Embedded Data
    - **TENANT** – Embedded Data
    - **ZONE** – Embedded Data

Note The data sources with a name in full **CAPS** will be used as part of the API call to BarTender Track & Trace.

The CompanyPrefix and ItemReference Named Data Sources should both reference the GTIN Database Field in the Product database as its data input.

For both the CompanyPrefix and ItemReference Named Data Sources, it is necessary to set validation rules via the Transforms tab to ensure the value is numeric, to truncate the entered 14-digit GTIN/PID so that only the wanted digits for the particular data source are kept, and to set the number of expected characters.



Example:
Assuming a standard PID of 14 digits as represented in the GTIN field of the data file, that in turn is constructed as a GTIN-14 for the RFID tag's input, whose parts consist of: **0 3663328 00007 6**

Indicator digit – 1 digit
Company Prefix – 7 digits
Item Reference – 6 digits
Check digit – 1 digit

- **Company Prefix rules**: Numeric only, truncate 1 from left and 6 from right, number of characters 7
- **Item Reference rules**: Numeric only, truncate 8 from left and 1 from right, number of characters 6

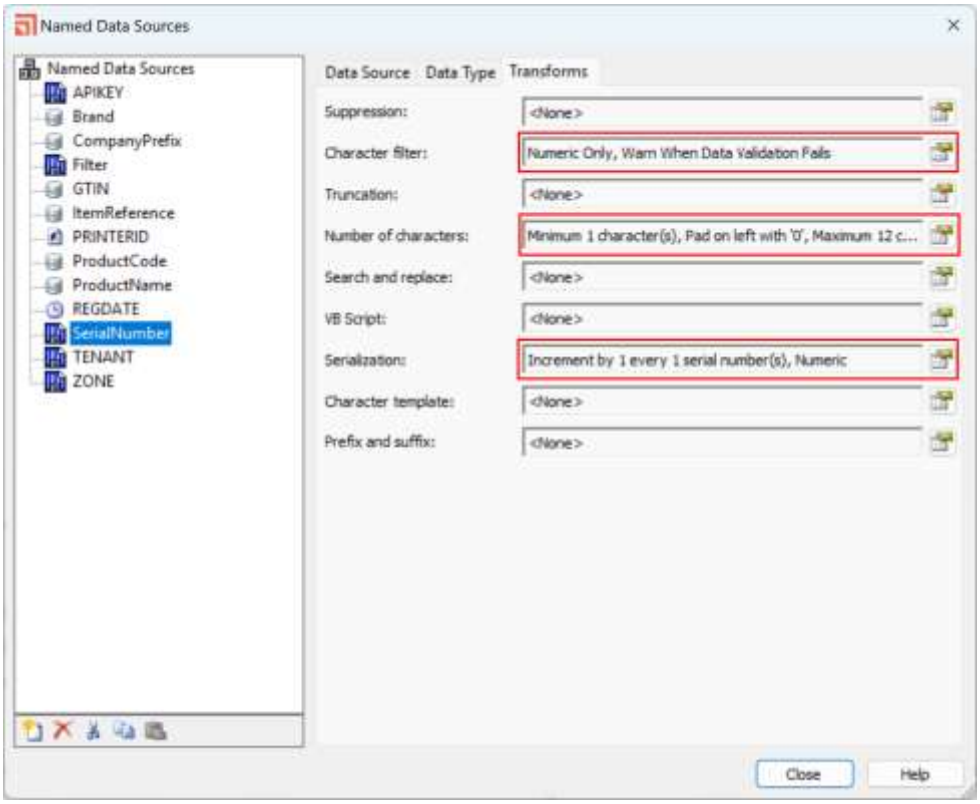
Note	The CompanyPrefix and ItemReference fields can vary in length, depending on the length of the other. However, between them the length in digits should equal 12. Refer to **Appendices C** and **F** for more information.

Serialization

The SerialNumber Named Data Source is the value that auto-increments with each successive label that's printed.

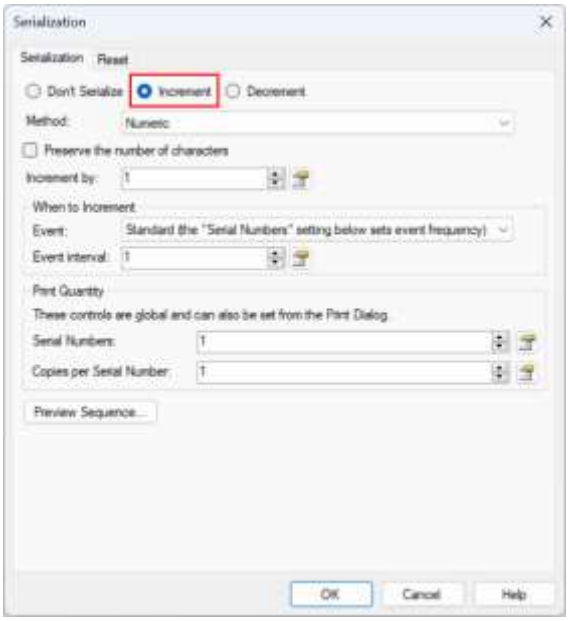
Under the Transforms tab you can set validation rules to:

- ensure only numeric data is used
- ensure that 1 – 12 digits minimum/maximum are set with padding “0” on the left
- serialize the value for each label printed.



The SerialNumber value is what serializes the RFID tag as part of the SGTIN-96 standard, ensuring that the tag’s encoded value is unique to correctly identify an instance of a product.

For the Serialization rule, increment the value by 1 for every 1 serialized label. Absolutely no duplicates are wanted.



By default, a serialized value is stored in the BarTender document file, and so will need to be saved between sessions for the next number in the sequence to be kept. An alternative to this is to use a Global Data Field, as the value is stored separately in the BarTender System Database. Another approach would be to use a serialization service via an HTTP request, via a third-party, though this is beyond the scope of this document.

Note	It is in the hands of the customer or partner to ensure that the serialized value used in the SGTIN-96 and resulting EPC HEXA value is maintained properly.

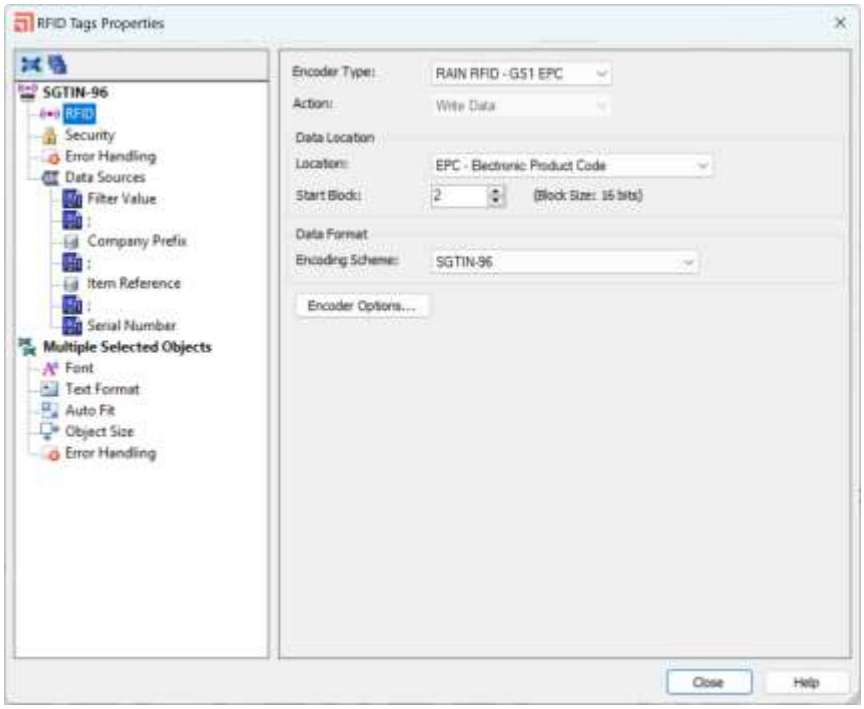
A future roadmap item for the platform would include a serialization service, accessed via an API call, that would provide per-product serialization while ensuring no duplicates.

ATTENTION: The APIs of the BarTender Track & Trace platform include a serialization service that can be called by the BarTender label software to generate a range of serial numbers on a per product basis without duplicates. Future versions of this instruction document will include documentation on this approach, plus implementation in the reference document BTW file.

The RFID tag

1. Add an RFID tag object (displayed as an icon adjacent to the label area) to the document of type RAIN RFID – GS1 EPC.
2. In the properties dialog of the RFID object, confirm that the Data Location is EPC – Electronic Product Code and Start Block is "2" respectively.

3. For the Encoding Scheme of the Data Format, choose SGTIN-96 from the dropdown list.



Once the SGTIN-96 is selected, the data source of the RFID object is split into component parts with a semi-colon (;) delimiter.

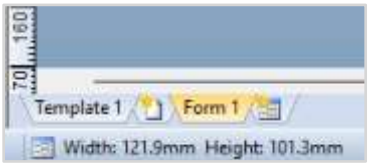
- Filter Value
- Company Prefix
- Item Reference
- Serial Number

Note that you'll need to replace each of the default data sources in the RFID object with the Named Data Sources that you already created in the earlier steps.

Once you're happy with your document layout and the data sources are all properly set-up, you can move on to defining an optional data entry form for the document.

Create a Form

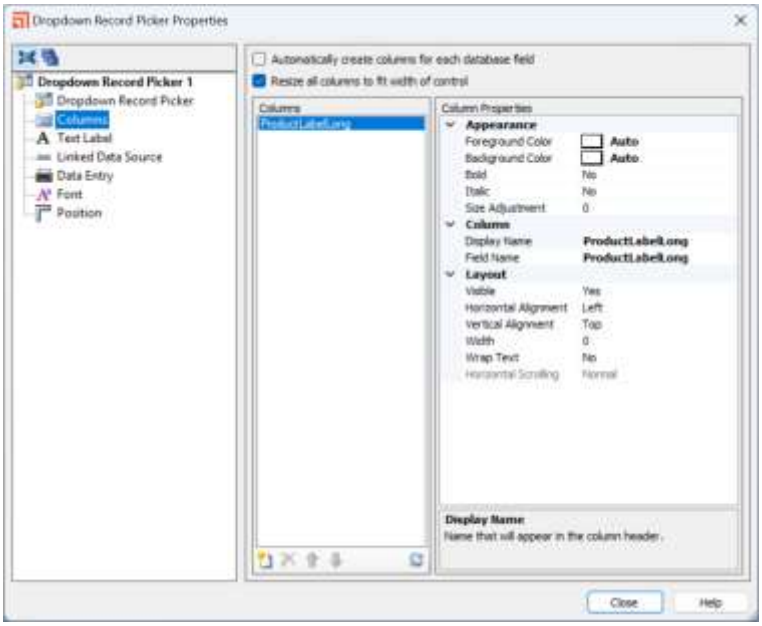
1. Click the Form tab, found at the bottom left of the design area.
2. Size the form to accommodate the form controls and other content that you wish to appear.



For this example, add a Dropdown Record Picker and Dropdown List form control.

- The Dropdown Record Picker enables the user to choose from the database the type of product for which they want to print/encode an RFID label.

It is usually best to constrict the displayed fields to those fields which display the name of the product to pick.



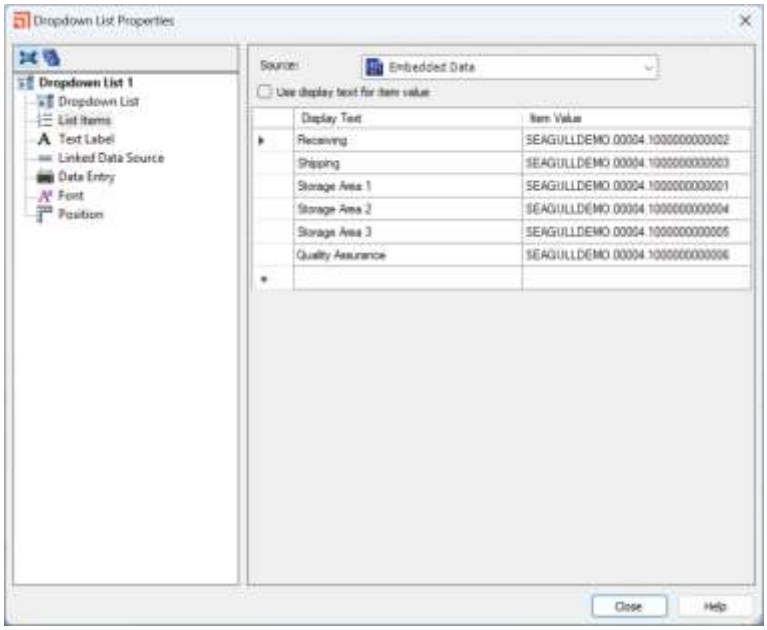
Dropdown Record Picker

- The Dropdown List is for displaying the various Zones into which the serialized instance of a product is to be delivered.

  Typically, this should be the Zone in which the printer is situated, and when the label is printed and applied to the physical item, it will already be tracked as being in that zone.

Be sure to Link the form control to the “ZONE” Named Data Source.

Note	The Display text will have the assigned name for the zone, while the value will set the Track & Trace code reference for the zone.

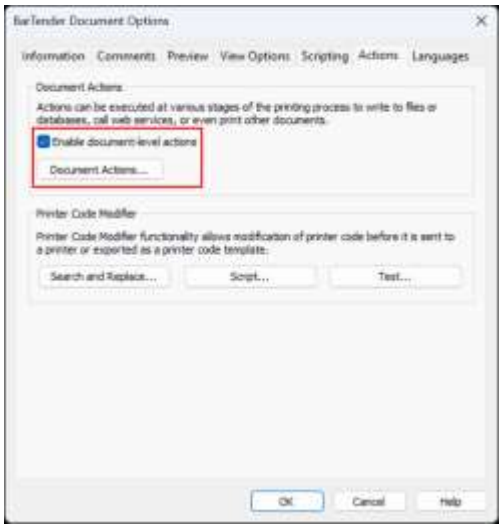


Dropdown List

Document Actions

With the label layout and data entry form completed we come to the Document Actions to execute upon pushing the identity up to the BarTender Track & Trace platform.

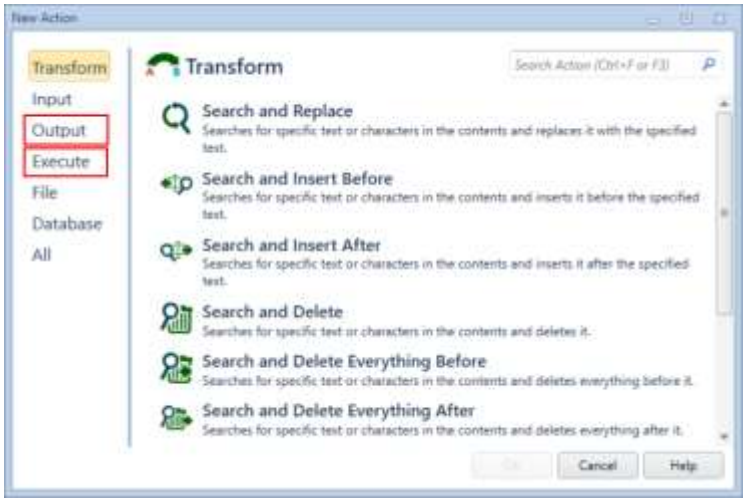
1. Select File > BarTender Document Options, and choose the Actions tab.
2. Tick the, **Enable document level actions** checkbox.
3. Click the **Document Actions** button.



4. In the Document Actions window that opens, add actions to the most appropriate document event in which you want the actions to execute.

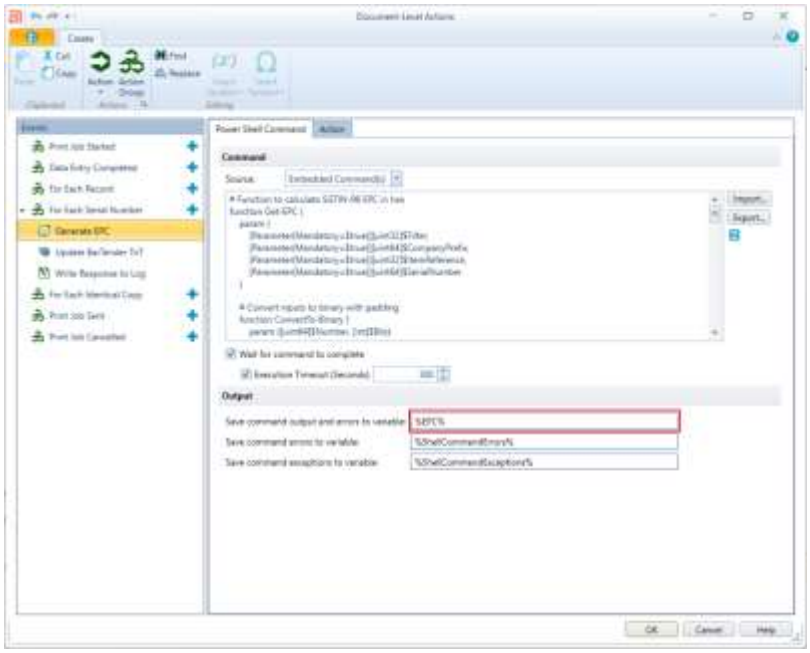
   In the case of our reference document this would be the For Each Serial Number event.

5. Click the **+** button for the desired Event.



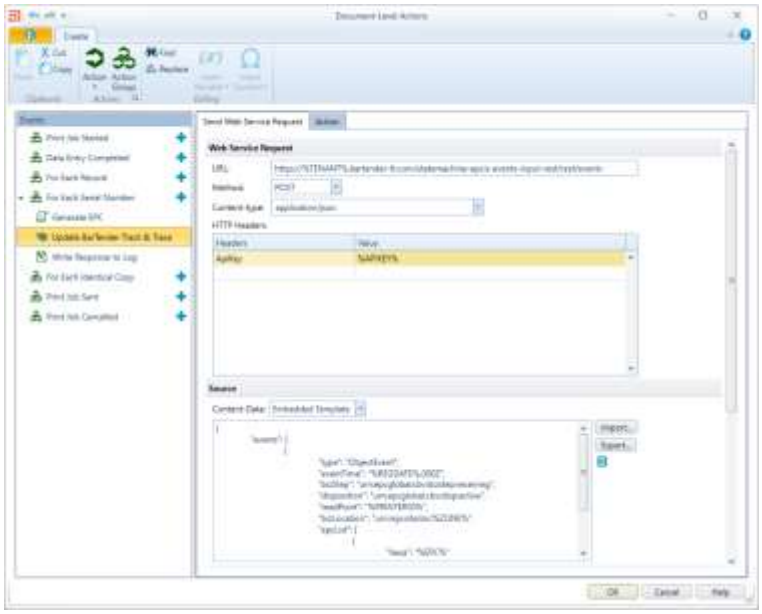
6. Create the following actions by Category > Action Name:
   - **Execute > Power Shell Command:** Executes a PowerShell script to generate the EPC HEXA from the supplied GTIN Field and Serial Number inputs.

     It is not currently possible to pass the value from the RFID object into a Named Data Source variable.



     Note that the output from this script is the variable **%EPC%** which will be used in the following actions. See the PowerShell script listed in **Appendix D** of this document or the example BTW file for the full set of code.

- **Output > Send Web Service Request:** Executes a Send Web Service Request to the BarTender Track & Trace platform. It uses the following Named Data Sources for dynamic usage:
    - %TENANT%
    - %APIKEY%
    - %REGDATE%
    - %PRINTERID%
    - %ZONE%
    - %EPC%



See the config and payload of this Send Web Service Request action in **Appendix E** of this document or the example BTW file for the full HTTP request.

- **Output > Write File (optional):** Use this action for client-side logging for traceability and diagnostics purposes. It is down to the user as to what information is logged, though certainly the following are recommended:
    - %EventDate%
    - %EventTime%
    - %ProductCode%
    - %EPC%
    - %CompanyPrefix%
    - %ItemReference%
    - %SerialNumber%
    - %ZONE%

Define mandatory:

- type: kept as “ObjectEvent”
- action: kept as “Observe”
- bizStep: kept as “urn:epcglobal:cbv:bizstep:receiving”

- disposition: kept as “urn:epcglobal:cbv:disp:active”

Define precisely:

- eventTime: Time when the EPC is encoded => Format of date for the encoding event is YYYY-MM-DDTHH:SS.MSZ (Sent as UTC time)
- bizLocation: Prefix of “urn:mjx:site:loc:” followed by the Zone/Fixture code as returned from the GET Zones request.
- epcList: A JSON array of hexadecimal EPC values to be commissioned. If you need to send as a URI, use the format as per this example: “epc”: “urn:epc:id:sgtin-96:3.3663328.000007.11235”

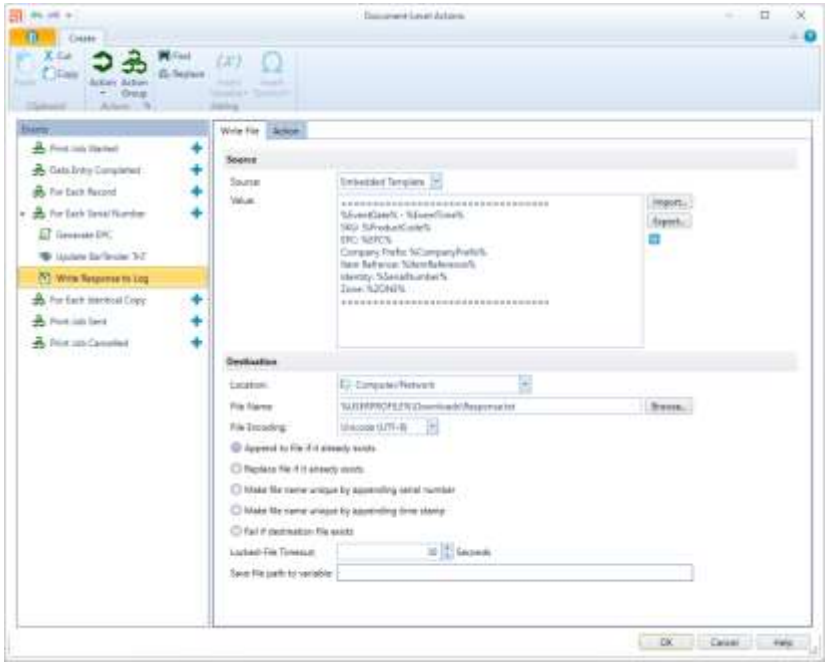
Define optional:

- readPoint: Optional attribute to identify production order line, e.g., Printer Name: “readPoint”: “Barcode Printer A”

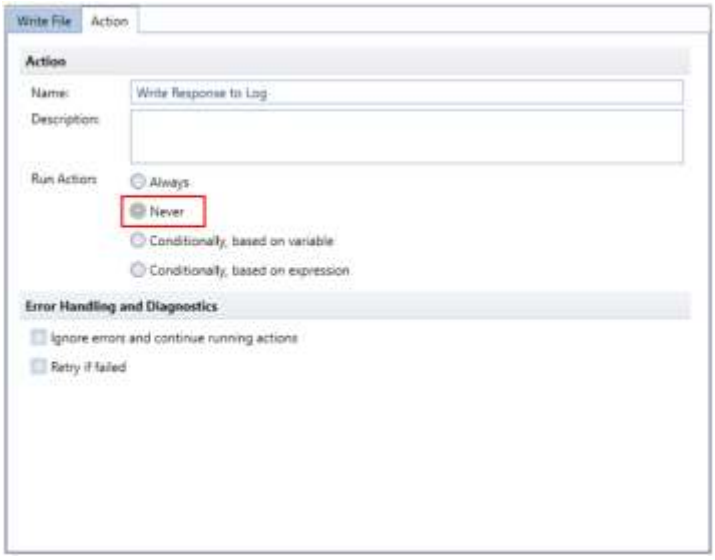
Responses:

- 204: Success
- 400: Input Error
- 404: Not found

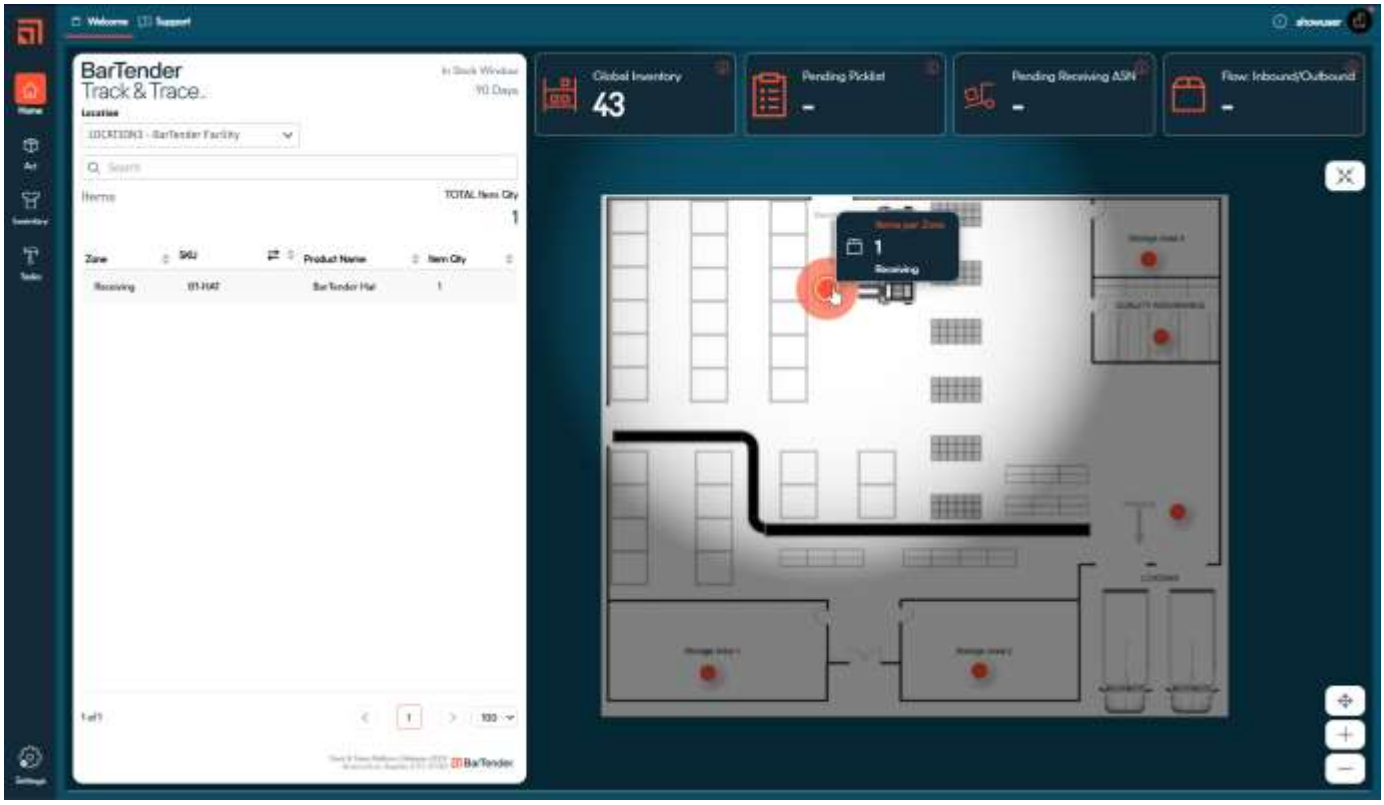
Note	There is no body to the response, only the HTTP code to indicate the result of the request.



An Action can be disabled, or made conditional, under the Action tab of the action, setting the Run Action as desired.

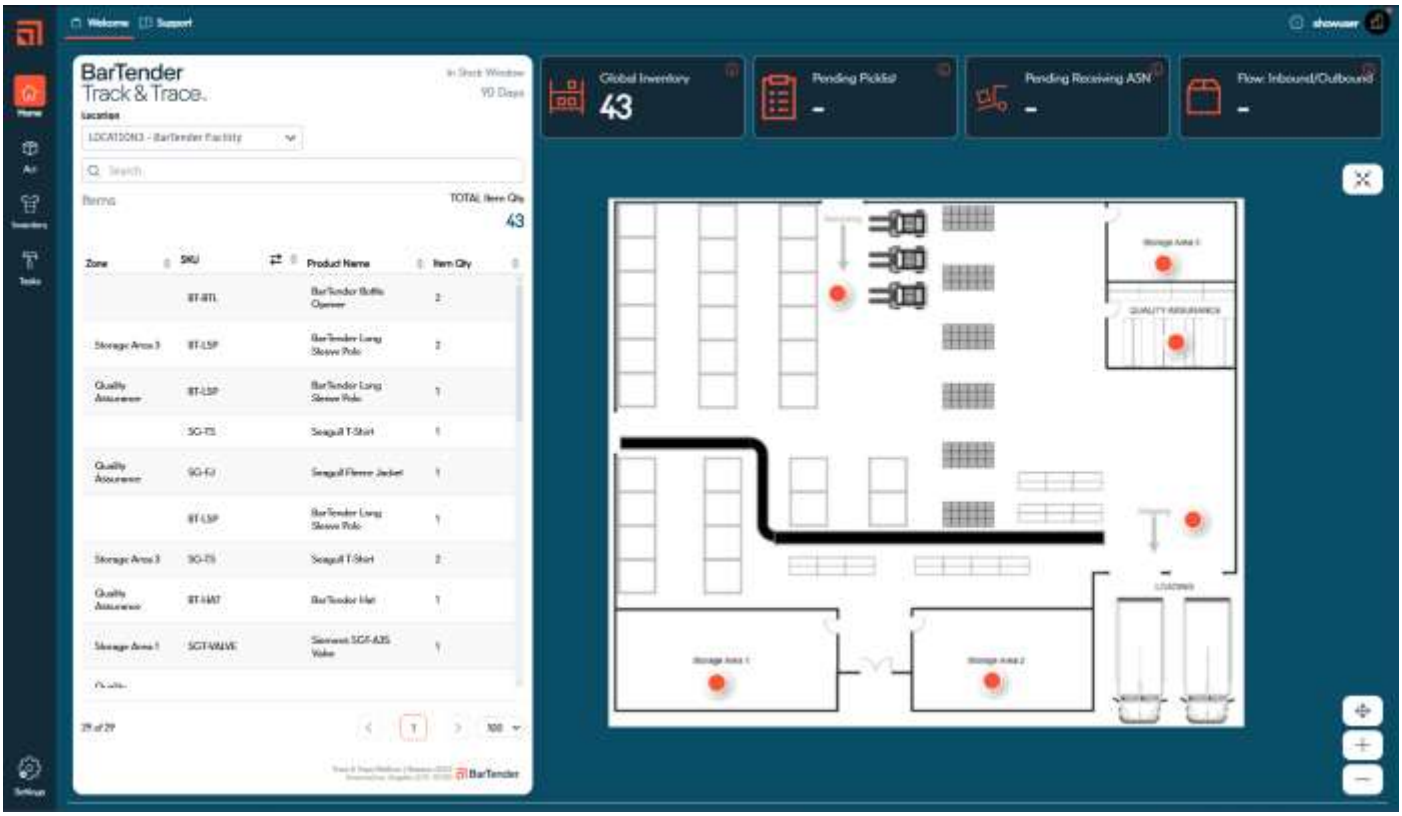


All being well, upon a quick refresh, there will be a new instance of a product received into the selected zone that has the same EPC (PID) value as the label just printed and applied to the physical item.

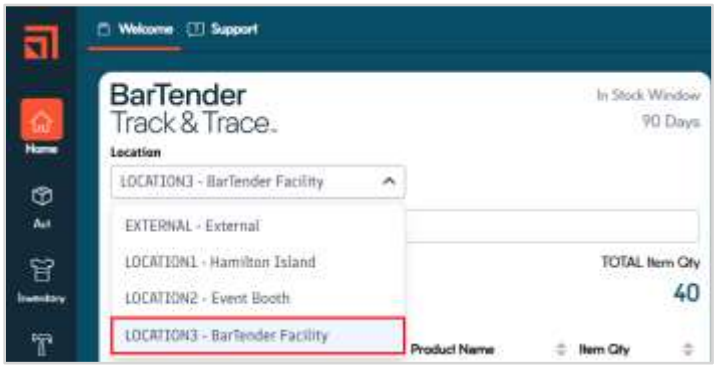


Appendix A: BarTender Track & Trace Web Console

Whereas the **Android** and **iOS** mobile apps are used to perform operational transactions of BarTender Track & Trace, the Web Console, accessed from your desktop web browser, provides a window onto your tenant for the purpose of dashboards, reports, and administration settings.



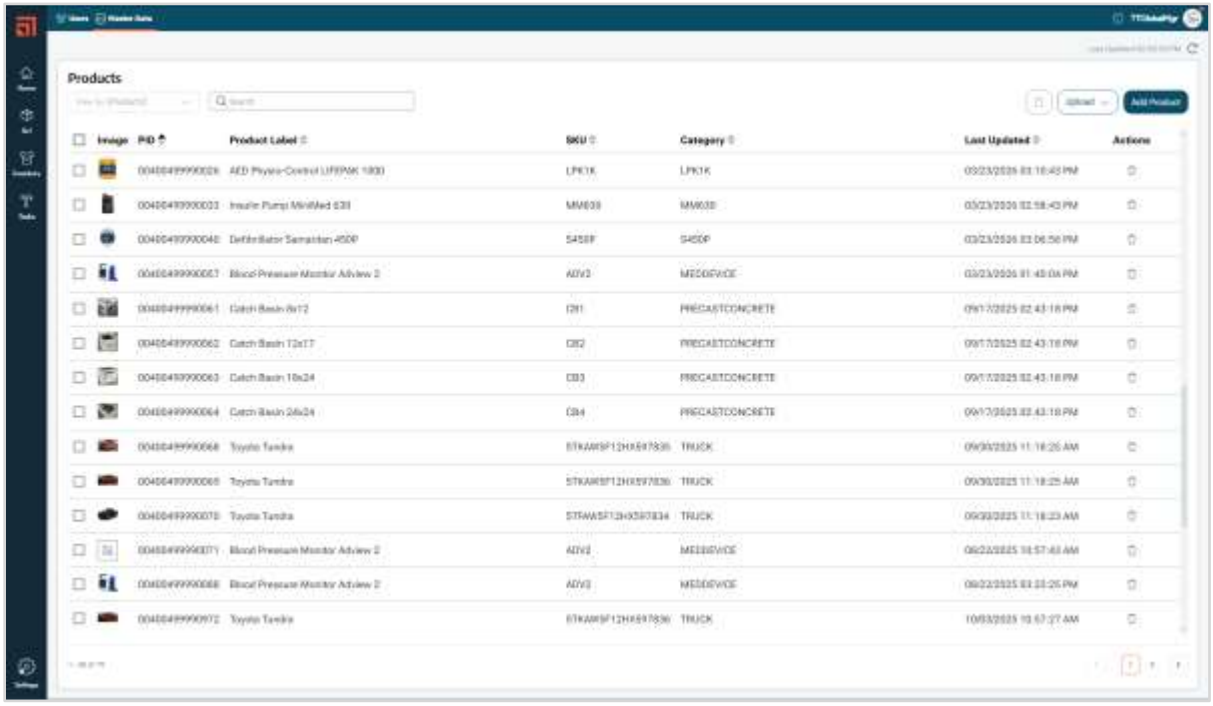
From the BarTender Track & Trace Home page a user has a global view of a physical location (factory/warehouse/office/store), and can easily switch between locations covered by the tenant. Use the Location dropdown list to change the current location.



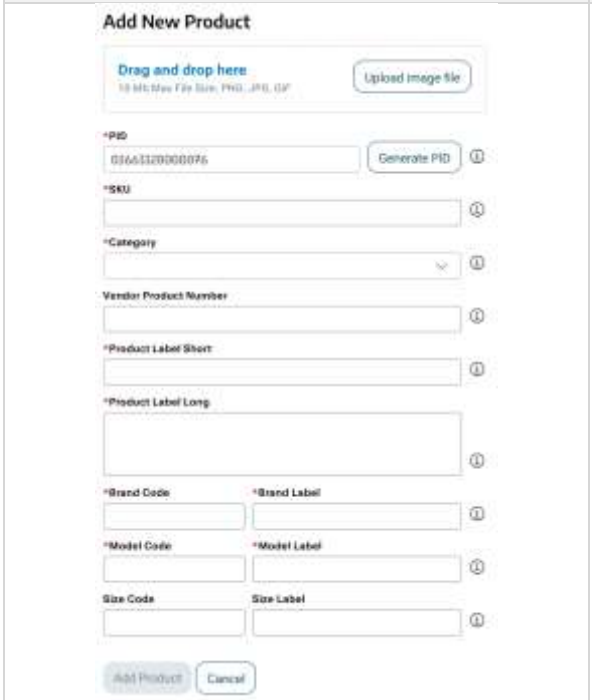
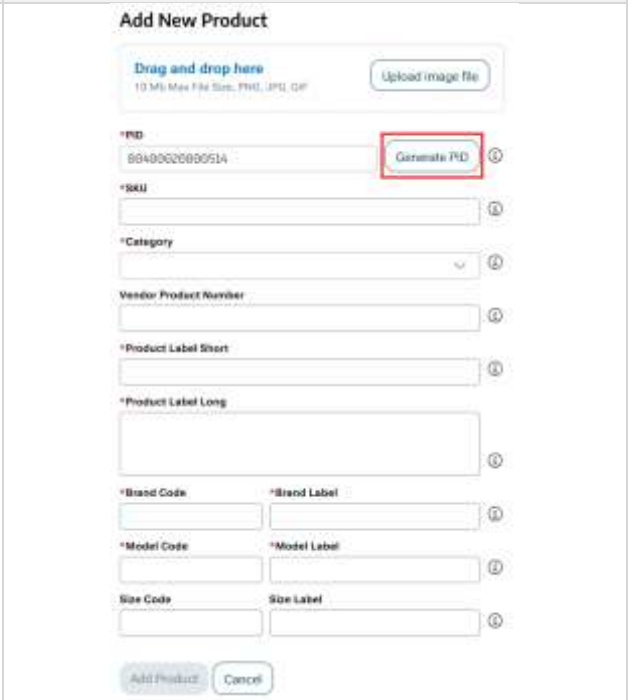
Use this powerful dashboard functionality to confirm the receipt of identities printed and pushed up to the platform by the BarTender label software.

Appendix B: Product Database

BarTender Track & Trace v12.1 added the ability for users to manage their own product database under the Master Data page (Settings > Track & Trace > Master Data).



A selected product's details can be modified, providing the user has adequate permission, or a new product can be added.

PID using your own Company Prefix and Item Reference	Auto-Generate a PID
	

The PID of a product in the current implementation is equal to that of a GTIN-14 and is used as the main identifier of a particular product type.

Appendix C: Barcodes to RFID encoding

The below screen capture of GTIN, GTIN-14, and GS1-EPC barcodes demonstrates how an SGTIN-96 RFID tag with encoded data is derived from more familiar GS1 barcodes:

Anatomy of a GS1 EPC

An SGTIN-96 is the most common RFID tag data format in the world. It consists of numeric digits for encoding a GTIN-14 and Serial Number as data.

Data Component	Description
T-bit	Set to "0" for GS1, and "1" for ISO as part of the Protocol Control bits (PC Word).
Header	Defines the scheme type — in this case SGTIN-96 (8 bits).
Filter	Specifies what the tag is being used for, which helps the reader to discriminate the type of unit being tracked.
Partition	Defines the length of the company prefix and by implication the item reference using the partition lookup table. **Note:** If you get the partition value wrong, then your tag will identify the incorrect product from the wrong manufacturer.
Company Prefix	Identifies the manufacturer/vendor. The international version (the GS1 global company prefix) has to be used, which means for a US company a "0" prefix needs to be added.
Indicator Digit	A GS1 standard from barcode — not RFID specific. Indicates the pack level of the item, where the meanings are defined by the company and their trading partners.
Item Reference	Identifies the type of item itself. This will correspond to a product as found in the product database of the BarTender Track & Trace Platform.
Serial Number	Unique number per tag — 12 digits (numeric) max. Allows for up to 274,877,906,944 unique serial numbers per GTIN.

What is the Indicator Digit?

The first digit of a GTIN-14, the indicator digit, indicates the pack level of the item being tracked. For example, is it a single item, a case, carton, pack, pallet of the same item just with a varying number of single items contained within?

Indicator Digit Table (0–8):

Indicator Digit	Typical Use Case / Meaning	Common Examples	Notes
0	No packaging hierarchy / Single unit or base trade item — often a zero padded GTIN-12 or GTIN-13	Consumer unit (e.g., individual bottle, box, or retail item); mixed packs/assortments; cases where the contained items have different GTINs	Most common for "each" or when there is no fixed hierarchy. Also used for standalone shipping containers that are not part of a multi-level hierarchy.
1	Lowest / Innermost packaging level above the consumer unit	Inner pack, tray pack, or smallest case (e.g., 6 or 12 units)	Often the first hierarchical level. Many companies start their hierarchy here.
2	Next packaging level (one level above 1)	Mid-level case or outer carton (e.g., 4 inner packs)	Builds the hierarchy upward.
3	Next packaging level (one level above 2)	Larger case or master pack	Continues the hierarchy.
4	Next packaging level (one level above 3)	Bulk case or pallet layer	Mid-to-higher logistics level.
5	Next packaging level (one level above 4)	Larger shipping case	Flexible assignment.
6	Next packaging level (one level above 5)	Outer carton or intermediate pallet load	Flexible assignment.
7	Next packaging level (one level above 6)	Near-outer packaging level	Flexible assignment.
8	Highest fixed packaging level (often the outermost before pallet)	Master carton, full pallet load, or largest fixed grouping	Frequently used for the top fixed-quantity case or pallet tier (before SSCC pallet labels).

How are these components used in a barcode and RFID tag?

Component	Example Value
GTIN-13	**3663328 00007 6**
GTIN-14	**0 3663328 00007 6**
SGTIN-96: Tag encoding schema	**urn:epc:tag:sgtin-96:3.3663328.000007.11235**
Filter value	**3**
Company Prefix	**3663328**
Indicator Digit	**0**
Item Reference	**00007**
Serial Number	**11235**

Note When using an encoding for private usage, you should usually set the indicator digit to a value of “0” for no packaging hierarchy.

Appendix D: Generating the EPC HEXA value

For reference purposes, below is the full PowerShell Script to calculate the SGTIN-96 HEXA value:

Note The output of the PowerShell script in the BarTender Action is to the variable **%EPC%** which is then used in the subsequent Send Web Service Request Action.

```
# Function to calculate SGTIN-96 EPC in hex
function Get-EPC {
    param (
        [Parameter(Mandatory=$true)][uint32]$Filter,
        [Parameter(Mandatory=$true)][uint64]$CompanyPrefix,
        [Parameter(Mandatory=$true)][uint32]$ItemReference,
        [Parameter(Mandatory=$true)][uint64]$SerialNumber
    )

    # Convert inputs to binary with padding
    function ConvertTo-Binary {
        param ([uint64]$Number, [int]$Bits)
        $bin = [Convert]::ToString($Number, 2)
        return $bin.PadLeft($Bits, '0')
    }

    # Construct binary EPC
    $headerBin = "00110000" # SGTIN-96 header (0x30)
    $filterBin = ConvertTo-Binary -Number $Filter -Bits 3

    switch ($cpLength) {
        12 {$partitionBin = "000"
            $companyPrefixBin = ConvertTo-Binary -Number $CompanyPrefix -Bits 40
            $itemReferenceBin = ConvertTo-Binary -Number $ItemReference -Bits 4
            } # Partition value 0 for 40-bit company prefix

        11 {$partitionBin = "001"
            $companyPrefixBin = ConvertTo-Binary -Number $CompanyPrefix -Bits 37
            $itemReferenceBin = ConvertTo-Binary -Number $ItemReference -Bits 7
            } # Partition value 1 for 37-bit company prefix

        10 {$partitionBin = "010"
            $companyPrefixBin = ConvertTo-Binary -Number $CompanyPrefix -Bits 34
            $itemReferenceBin = ConvertTo-Binary -Number $ItemReference -Bits 10
            } # Partition value 2 for 34-bit company prefix

        9 {$partitionBin = "011"
           $companyPrefixBin = ConvertTo-Binary -Number $CompanyPrefix -Bits 30
           $itemReferenceBin = ConvertTo-Binary -Number $ItemReference -Bits 14
           } # Partition value 3 for 30-bit company prefix

        8 {$partitionBin = "100"
           $companyPrefixBin = ConvertTo-Binary -Number $CompanyPrefix -Bits 27
           $itemReferenceBin = ConvertTo-Binary -Number $ItemReference -Bits 17
           } # Partition value 4 for 27-bit company prefix

        7 {$partitionBin = "101"
           $companyPrefixBin = ConvertTo-Binary -Number $CompanyPrefix -Bits 24
           $itemReferenceBin = ConvertTo-Binary -Number $ItemReference -Bits 20
           } # Partition value 5 for 24-bit company prefix
```

```
        6 {$partitionBin = "110"
           $companyPrefixBin = ConvertTo-Binary -Number $CompanyPrefix -Bits 20
           $itemReferenceBin = ConvertTo-Binary -Number $ItemReference -Bits 24
           } # Partition value 6 for 20-bit company prefix
    }

    $serialNumberBin = ConvertTo-Binary -Number $SerialNumber -Bits 38

    $binaryEPC = $headerBin + $filterBin + $partitionBin + $companyPrefixBin +
$itemReferenceBin + $serialNumberBin

    # Verify length
    if ($binaryEPC.Length -ne 96) {
        return "Error: Invalid EPC binary length"
    }

    # Convert binary to hex
    function ConvertTo-Hex {
        param ([string]$Binary)
        $hex = ""
        for ($i = 0; $i -lt $Binary.Length; $i += 4) {
            $chunk = $Binary.Substring($i, 4)
            $hex += [Convert]::ToString([Convert]::ToInt32($chunk, 2), 16).ToUpper()
        }
        return $hex
    }

    return ConvertTo-Hex -Binary $binaryEPC
}

# Input Values
$filter = "%Filter%"
$companyPrefix = "%CompanyPrefix%"
$itemReference = "%ItemReference%"
$serialNumber = "%SerialNumber%"
$cpLength = ($companyPrefix.length)
$irLength = ($itemReference.length)

$(Get-EPC -Filter $filter -CompanyPrefix $companyPrefix -ItemReference $itemReference
-SerialNumber $serialNumber)
```

Appendix E: API request to Track & Trace platform

The document Action to push the EPC identity up to the BarTender Track & Trace platform leverages the API of BarTender Track & Trace via a Send Web Service Request Action of BarTender.

ATTENTION: The APIs of the BarTender Track & Trace platform are undergoing a process of streamlining, simplifying, polishing, and fully documenting prior to official publication. As such, the API calls needed to perform this function will evolve in future updates, with the approach below becoming deprecated. Future versions of this instruction document will be updated and use the published API.

The **POST**ed HTTP request will have a URL and headers as below:

```
{{root}}/statemachine-epcis-events-input-rest/rest/events
Content-Type: application/json; charset=utf-8
Apikey: XXXXXXXXXXXXXXX
```

With the payload of the request as below:

```
{
    "events": [
        {
            "type": "ObjectEvent",
            "eventTime": "%REGDATE%.000Z",
            "bizStep": "urn:epcglobal:cbv:bizstep:receiving",
            "disposition": "urn:epcglobal:cbv:disp:active",
            "readPoint": "%PRINTERID%",
            "bizLocation": "urn:mjx:site:loc:%ZONE%",
            "epcList": [
                {
                    "hexa": "%EPC%"
                }
            ],
            "action": "OBSERVE"
        }
    ]
}
```

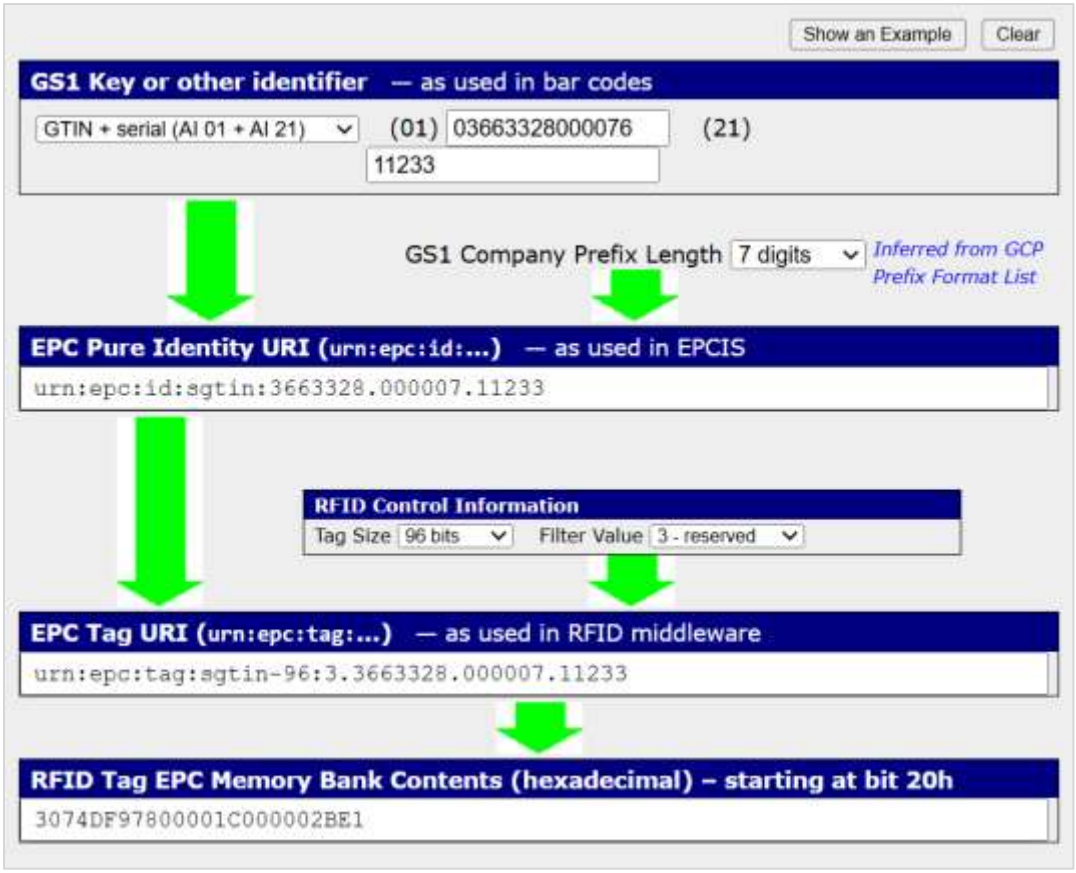
Note Named Data Source variable values are passed into the payload as denoted by the %VARIABLE% syntax. A successful request will simply return an HTTP 204 response code with no body.

Appendix F: Test with the GS1 Encoder / Decoder

Use this handy GS1 tool for double checking the EPC HEXA generated by BarTender from the GTIN-14 and Serial Number inputs provided at print/encode time:

https://www.gs1.org/services/epc-encoderdecoder

1. Select **GTIN + serial (AI 01 + AI 21)** from the dropdown list and enter in the GTIN-14 after the (01) application identifier.
2. After the (21) application identifier enter the Serial Number that identifies an instance of the product type.
3. Set the Company Prefix length. In our examples we used a length of 7 digits.
4. Tag Size should be set to 96 bits for an SGTIN-96.
5. Filter value is down to preference and use case. In our examples we used a filter value of 3.
6. Compare the EPC Tag URI and the EPC HEXA value as shown in the BarTender Designer, with what is generated by the GS1 tool.



Related Documentation

Technical Documents

To view and download technical documents, visit:

https://www.bartendersoftware.com/resources/library

User Guides

Support Articles

- Getting Started with Track & Trace
- Getting an Overview with the Track & Trace Homepage
- Setting Up the Track & Trace Mobile App
- Keep Track of Inventory Assets with Track & Trace
- Getting Started with BarTender Cloud

Documentation

- Reviewing Assets in Track & Trace
- Managing Track & Trace
- Working with Track & Trace
- BarTender Cloud Actions API

Other Resources

Please visit the BarTender website at https://www.bartendersoftware.com.