NiceLabel Automation Version 1.3

Release Notes

Rev-1405

Table of Contents

What is New in NiceLabel Automation?

New Label Template Features

RFID support

Automation 1.3 extends the existing label printing capabilities with support for the programming of the RFID tags.

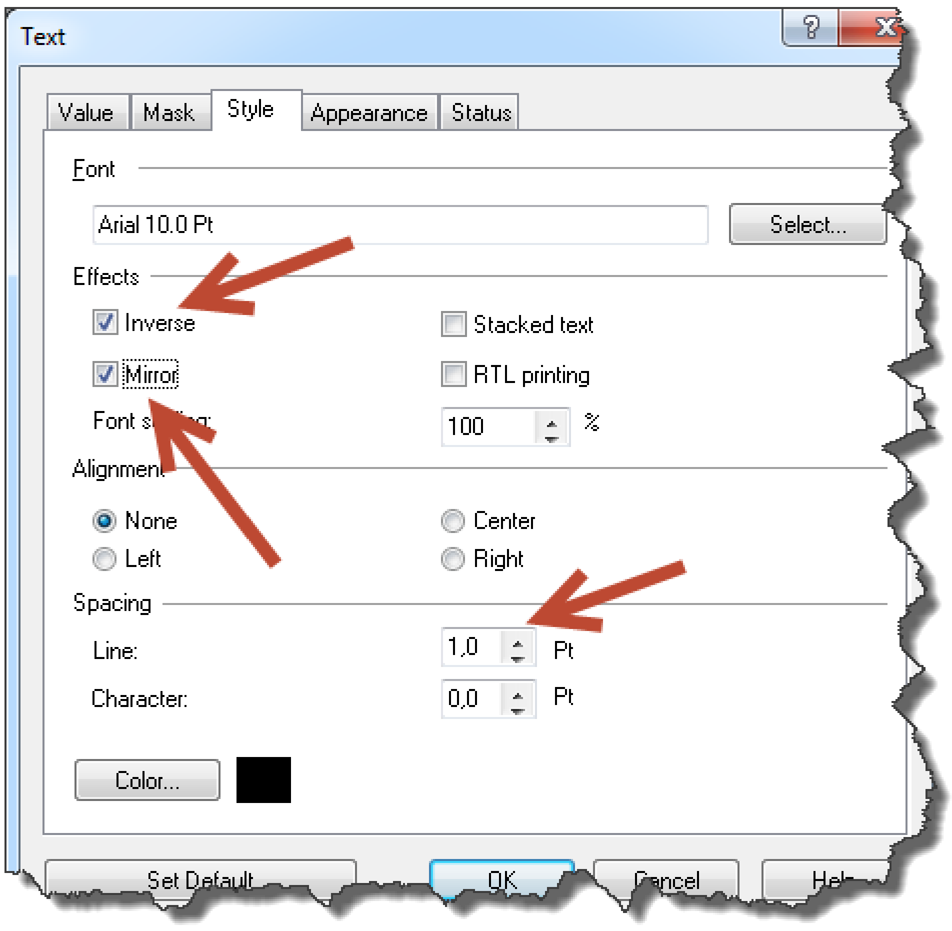
You can encode the RFID tags embedded in the smart-label and print the same label concurrently. The RFID support must be enabled in the label template during the design phase. The list of tags that you can use in the label template depends on the NiceLabel printer driver that is associated with the label. Different printer models will enable different RFID tags.

You can also use EPC-enabled tags.

Text object updates

Automation has been updated in order to support more design properties of the Text object.

- **Line spacing.** The amount of extra space (or less space) between lines of data.
- **Inverse.** The text is printed in inverse color. The white background becomes black, and the black font color becomes white.
- **Mirror.** The object is printed mirrored (in the vertical axis).



Text

Value | Mask | Style | Appearance | Status

Font

Arial 10.0 Pt — Select...

Effects

- [✓] Inverse
- [ ] Stacked text
- [✓] Mirror
- [ ] RTL printing
- Font scaling: 100 %

Alignment

- [● None]
- ○ Center
- ○ Left
- ○ Right

Spacing

- Line: 1,0 Pt
- Character: 0,0 Pt

Color... ■

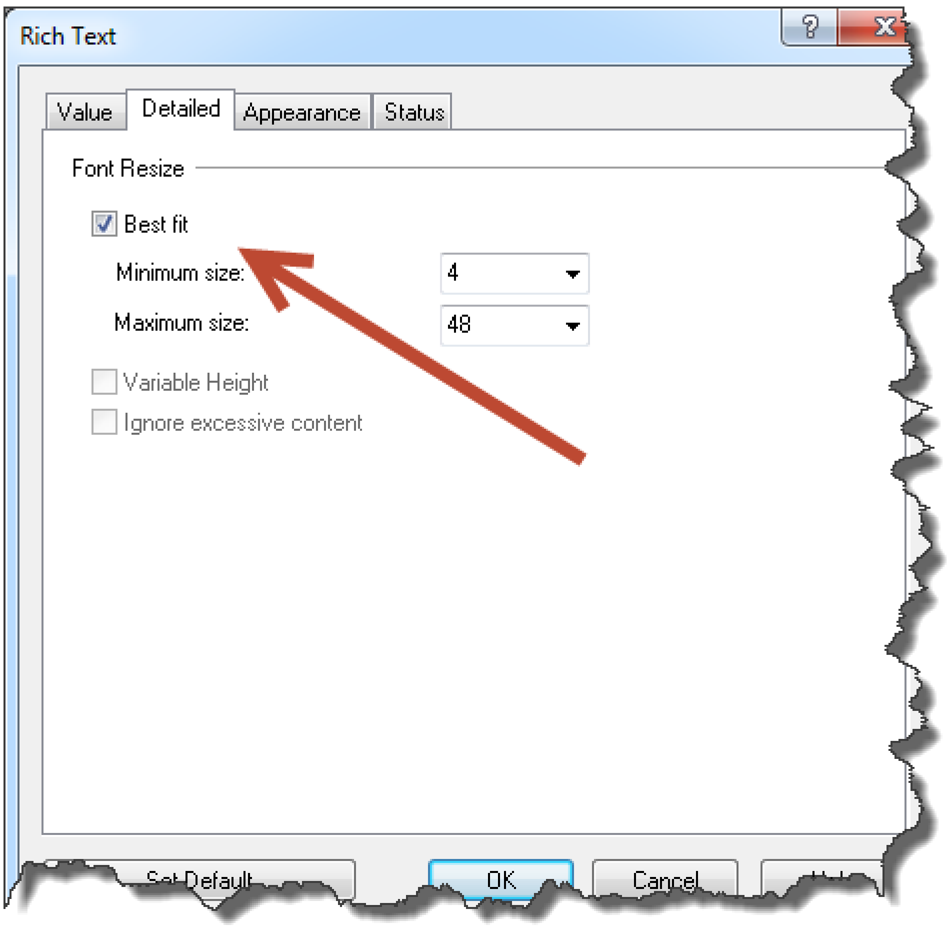
Figure 1: New formatting options supported for the Text object

Best fit support for a Rich Text object

Automation 1.3 supports the **Best fit** option for RTF objects.

When this option is enabled in the label template, the processing logic in NiceLabel Automation adjusts the point sizes of the text in order to fit all of the content into the object’s bounding box.

The point sizes are auto-determined between the user-defined minimum and maximum allowed values.


Rich Text

Value | Detailed | Appearance | Status

Font Resize

[✓] Best fit

Minimum size: 4

Maximum size: 48

[ ] Variable Height

[ ] Ignore excessive content

Figure 2: Support for the Best fit option for an RTF object

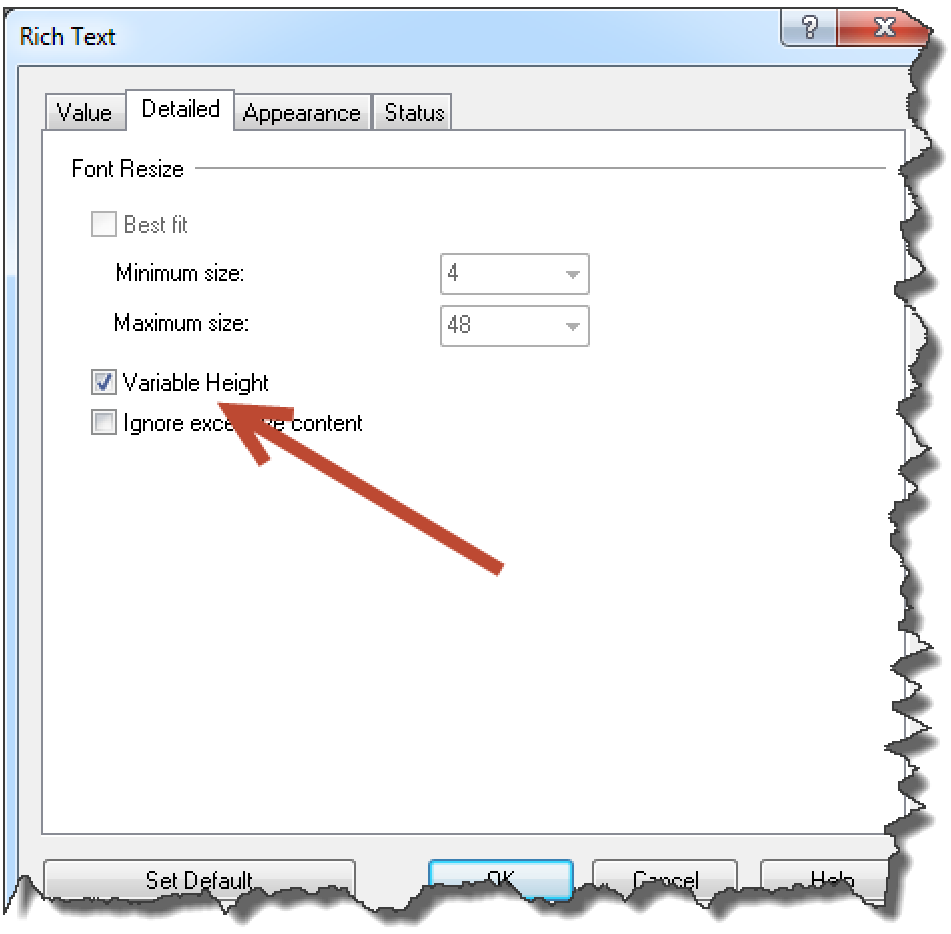
Variable height support for a Rich Text object

Automation 1.3 supports the **Variable height** option for RTF objects.

When this option is enabled in the label template, the processing logic in Automation adjusts the height of the object in order to fit all of the content without changing the point sizes.

If more data is provided, the object's vertical size will grow.

If less data is provided, the object's vertical size will shrink.


Rich Text

Value | Detailed | Appearance | Status

Font Resize

- [ ] Best fit
- Minimum size: 4
- Maximum size: 48
- [✓] Variable Height
- [ ] Ignore excessive content

Figure 3: Support for Variable height option for the RTF object

Full justification support for an RTF object

Automation 1.3 supports the **Full justification** option for RTF objects.

When you define the RTF object in the label designer and enable the full justification option, the text will be aligned along both the left and right margins. In justified text, the spaces between words (or characters to a lesser extent) are stretched or sometimes compressed in order to make the text align with both the left and right margins.

Note: you need the application NiceLabel Pro V6.3 to enable full justification in the RTF object.

RESTful Web Services (Updates to HTTP Trigger)

Support for multiple triggers running on the same port

You can define multiple HTTP triggers running on the same port. To identify one trigger from another, you can define the "path" parameter. The trigger will fire when the HTTP request's socket and URL path are matched.

This functionality enables Automation to expose multiple HTTP triggers on the same socket and define the RESTful Web Services. The client will use many triggers (each responsible for a certain operation) through a single socket in a REST like syntax, causing different triggers to be fired by different URLs.

Support for additional HTTP header fields

Some HTTP servers (RESTful services) might require custom HTTP header fields to be included in the HTTP request from the clients. The HTTP header fields are components of the HTTP message header and define the operating parameters of an HTTP transaction.

By default, Automation generates an HTTP message with only mandatory HTTP header fields. When NiceLabel Automation is used as a client requesting (or providing) data from remote HTTP servers, you can define any additional custom HTTP header fields that the server requires.

```
Accept: application/json; charset=utf-8
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/31.0.1650.63 Safari/537.36
Content-Type: application/json; charset=UTF-8
```

Figure 4: Examples of the HTTP header fields "Accept", "User-Agent", and "Content-Type"

Automatic parsing of field:value pairs in the query string

When you define a HTTP trigger in Automation, you usually provide the data **in the body of the HTTP request** and then extract the data using filters.

In Automation 1.3, you can also provide the data **in the HTTP query string**.

```
http://server/path/?query_string
```

Figure 5: A typical URL containing a query string

The data provided in the query string must be in the pairs of the variable's (field's) name and variable's value.

```
http://server/path/?variable1=value1&variable2=value2&variable3=value3
```

Figure 6: The query string providing values for three variables: variable1, variable2, and variable3

The HTTP trigger has the built-in support to extract the values of all the fields and store them into the trigger variables of the same name. For the above example, you would define the trigger variables with names `field1`, `field2`, and `field3` and the trigger itself would assign the values `value1`, `value2`, and `value3` to them, respectively, without any configuration.

Web Service (SOAP) Updates

Support for the basic authentication for Web Service action

Web Service action allows NiceLabel Automation to connect to the remote Web Service and executes its methods. Based on the input parameters, the Web Service provides the result.

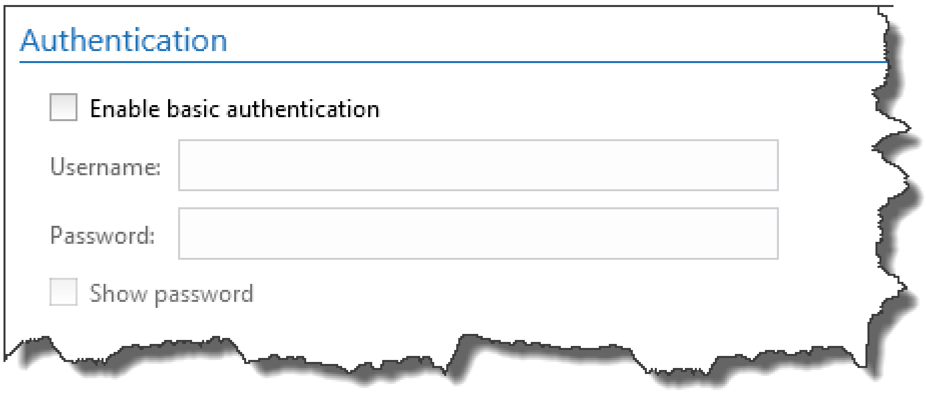


Figure 7: Provide user credentials for connection to the remote Web Service

When configuring the Web Service action, you can enter the user credentials for the HTTP basic authentication and connect to the Web Services that require authentication.

Automatic parsing of field:value pairs in the Web Service trigger

The Web Service trigger exposes new methods that allow users to assign values to the variables without the need to define any filter.

The field:value pairs can be provided in one of the following structures. The trigger will auto-identify which one has been delivered.

```
<Variables>
  <variable name="Variable1">Value1</variable>
  <variable name="Variable2">Value2</variable>
  <variable name="Variable3">Value3</variable>
</Variables>
```

Figure 8: Providing the field:value data in the XML structure

```
Variable1=Value1
Variable2=Value2
Variable3=Value3
```

Figure 9: Providing the field:value data in plain text

The Web Service trigger has the built-in support to extract the values of all the fields and store them in the trigger variables of the same name. For the above example, you would define the trigger variables with the names `variable1`, `variable2`, and `variable3` and the trigger itself would assign the values `value1`, `value2`, and `value3` to them, respectively, without any configuration.

Other Updates

Applying printer settings to the label

Imagine a situation where you have to print the same label template to various label printers. Each printer might require slightly different changes in its configuration, such as the speed, darkness, and offsets. The user who is in charge of such scenarios usually creates copies of labels and optimizes each label for the associated label printer.

Each label prints correctly to the target printer, but this approach is a nightmare from the maintenance point of view. You have to manage the same label design, yet it is saved in many label templates (.LBL files).

You could already use the **Set Printer Parameter** action in NiceLabel Automation Pro and Automation Enterprise to fine-tune the basic parameters, such as speed, darkness, and offsets.

Now, the new version of Automation software allows you to fine-tune **each and every setting** in the printer driver. In Windows, the printer driver settings for every printer installed in your system are saved in the internal Windows structure, known as DEVMODE. You can extract the DEVMODE on one computer and apply it to the label just before printing in the production environment.

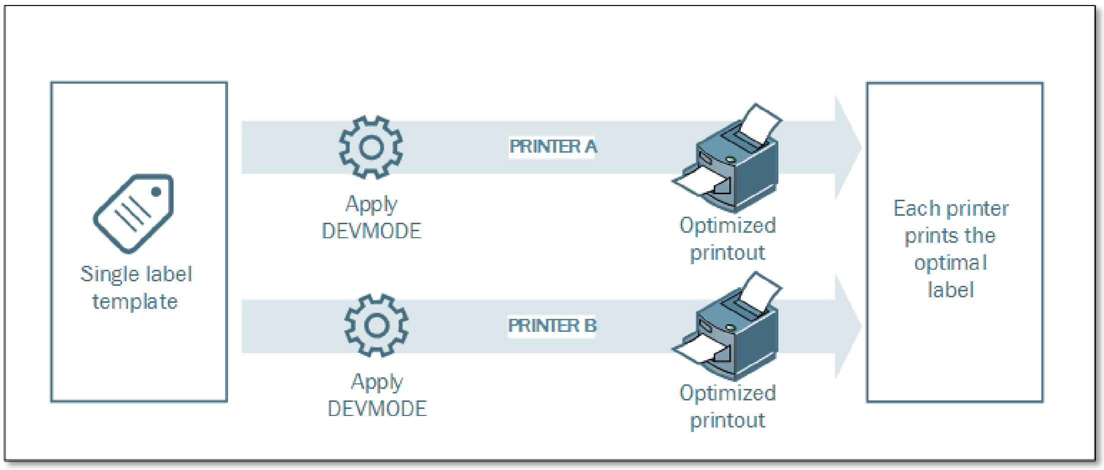


Figure 10: Applying different DEVMODE to the same label provides optimal printout to various printers

The typical label design & print process would be:

1. Just one label template is designed for a particular need, such as “shipping label” or “product label”.
2. Test printing is performed for every printer used in the label production.
3. During testing, the optimal driver settings are defined and saved in the driver, including speed, darkness, offsets, printing mode, graphics encoding options, dithering, and other.
4. The DEVMODE is extracted from the driver settings in the test environment.
5. The DEMODE is applied to the label in the production environment.

The result is an optimal printout of the single label template to any of the production printers.

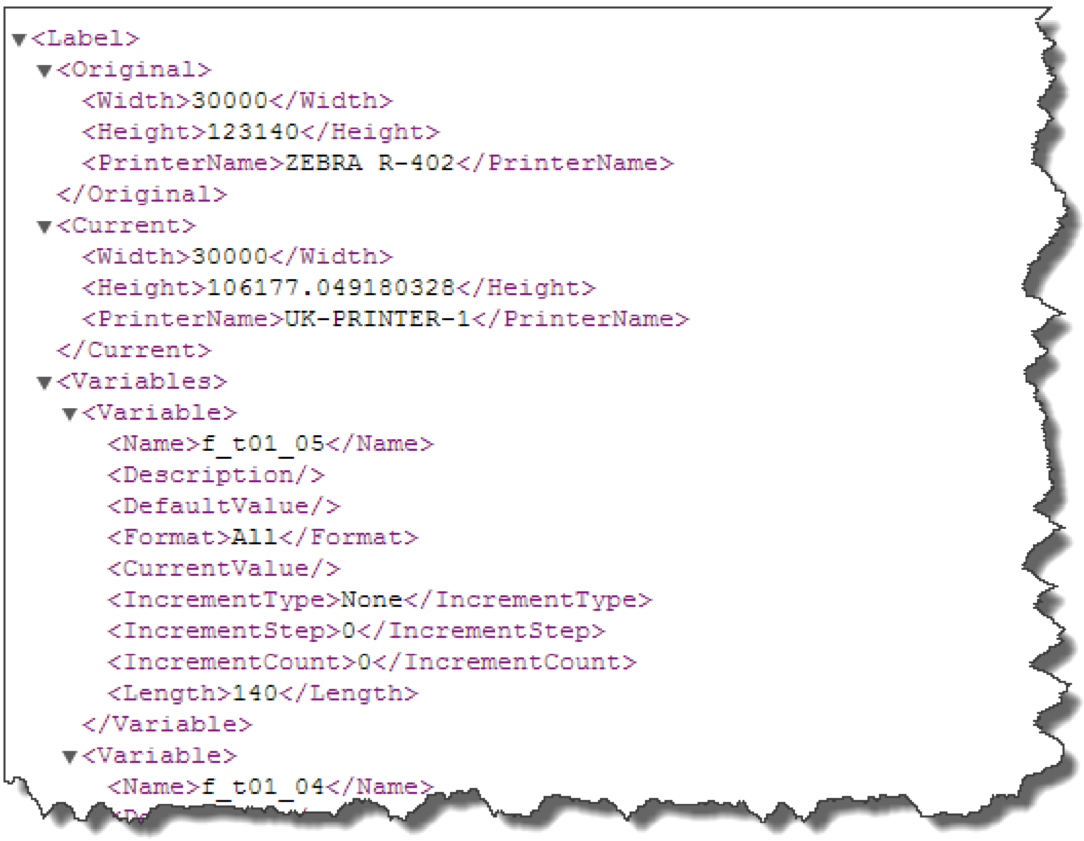
Available only in Automation Pro and Automation Enterprise.

Get label information action

The new action **Get label information** generates an XML file with a description of the requested label.

The first section in the returned data contains label design-related information, such as the label width and height and the name of printer used during the design and for the printing.

The second section contains a list of variables and their major properties, such as name, type, and current value.



```
▼<Label>
  ▼<Original>
     <Width>30000</Width>
     <Height>123140</Height>
     <PrinterName>ZEBRA R-402</PrinterName>
   </Original>
  ▼<Current>
     <Width>30000</Width>
     <Height>106177.049180328</Height>
     <PrinterName>UK-PRINTER-1</PrinterName>
   </Current>
  ▼<Variables>
    ▼<Variable>
       <Name>f_t01_05</Name>
       <Description/>
       <DefaultValue/>
       <Format>All</Format>
       <CurrentValue/>
       <IncrementType>None</IncrementType>
       <IncrementStep>0</IncrementStep>
       <IncrementCount>0</IncrementCount>
       <Length>140</Length>
     </Variable>
    ▼<Variable>
       <Name>f_t01_04</Name>
```

Figure 11: Label described in an XML structure

The new action provides for new opportunities for scenarios such as:

- **Getting a list of label variables.** If you integrate Automation into some existing application as the label print-engine service, you will have the possibility to retrieve a list of label-defined variables so that your application can provide their values.
- **Track consumption of consumables.** If your labels adjust their length based on the amount of the incoming data (which is a common request in the textile & garment industry), you will welcome the possibility for feedback regarding the actual label length. Based on this information, you will know exactly how much media has been used and when the print user needs to restock, so you can issue an advanced warning and make sure the customers are never out of consumables.
- **Keeping local information of the printed data.** The returned XML files can be used as a log repository or proof-of-data-printing.

Available only in Automation Enterprise.

Support for in-database-stored images (BLOB fields)

NiceLabel Automation 1.3 supports the images that have been saved in a database together with the rest of the product information. Some database types allow you to save large binary data into fields, known as binary large object (BLOB) fields.

To be able to use the in-database-stored images, you have to configure the database connection in the label template and provide the BLOB field value to the image object.

NiceLabel Automation will print the received binary data as the image object.

Encoding special characters (control codes)

The option to encode special characters has been added to the **String Manipulation** action and filters. The special characters (or control codes) are characters not available on the keyboard, such as Carriage Return or Line Feed. NiceLabel Automation uses a notation to encode such characters in human-readable form, such as <CR> for Carriage Return and <LF> for Line Feed.

The new option converts the special characters from NiceLabel syntax into actual binary characters.

For example: when you receive the data "<CR><LF>", NiceLabel Automation will use it as plain string of 8 characters. To use the received data as two binary characters **CR** (Carriage Return - ASCII code 13) and **LF** (Line Feed - ASCII code 10), enable this new option.

Optimization in caching of the remote files

To speed up label printing, NiceLabel Automation can already create a local cache of files that exist on the remote servers and keep it up to date.

The updated caching algorithm in NiceLabel Automation 1.3 provides a significant speed improvement in cases, when the dependent files are not known in advanced, but became recognized during the print processing phase.

A typical example is a list of images that must print on the label. The trigger event might provide the label name and product ID. The image names to be used with that label are read from the associated database and become known only during the processing.

Automation will now also cache such dependent files for the next time they are requested.

Available only in Automation Enterprise.

Optimized logging to the Control Center

Logging to centralized Control Center database has been optimized to offer substantial performance improvements as well as using less system resources on production Automation servers.

Available only in Automation Pro and Automation Enterprise.

Updated Application Identifiers for GS1 support

NiceLabel Automation has been updated with the changes from the GS1 General Specifications Version 14, January 2014.

New GS1 Application Identifiers are supported: AI(16) Sell by date, AI(713) National Healthcare Reimbursement Number (NHRN) - Brasil DRN, AI(8010) Component / Part Identifier (CPID) and AI(8011) Component / Part Identifier Serial Number (CPID SERIAL). AI(8007) International Bank Account Number (IBAN) has updated length of data.

Note: you need the application NiceLabel Pro V6.3 to be able to design a label with new Application Identifiers.