Tricolops Dimensioning Application Integration Manual

Target: v1.8.1 or higher
Revision: 1.6
September 14, 2021

Table of Contents

Customizing Hotkey Integration

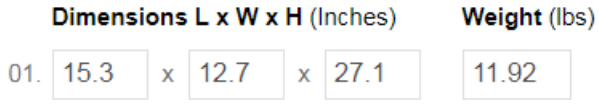
It is recommended to use the hotkey integration method if you cannot communicate with the Tricolops Dimensioner programmatically.

After the specified hotkey is pressed, keyboard strokes are simulated to enter dimensions and / or weight into web based or Windows applications. The sequence can be customized by going to Advanced – Settings – Advanced – Hotkey Sequence.

The hotkey strokes available are:

Character	Key sequences sent
A	Ctrl – A, used to highlight all characters in a text field
B	Backspace, used to remove characters in a text field if Ctrl-A is not available.
E	Enter, used to submitting a form.
S	Space
T	Tab, used to switch from one text field to the next
l	Length of the package rounded (without decimal places)
L	Length of the package (with decimal places)
w	Width of the package rounded (without decimal places)
W	Width of the package (with decimal places)
h	Height of the package rounded (without decimal places)
H	Height of the package (with decimal places)
m	Weight of the package rounded (without decimal places)
M	Weight of the package (with decimal places)

Additionally, the keystrokes can be customized based on the name of the window. The following table demonstrate some examples:

Sequence	Behaviour
ALTWTHTM	 Highlights the current textbox, then sends exact length, width, height, and weight on all windows
AMTLTWTH	Highlights the current textbox, then sends the exact weight, length, width, and height on all windows
MTTTlTwTh	Inputs the exact weight, tabs three times, then sends rounded length, width, and height on all windows (For UPS WorldShip)
ups,MTTTlTwTh, netparcel,ALTWTHTM, AlTwThTM	Sends keystrokes MTTTlTwTh into all windows containing the name ups Sends keystrokes ALTWTHTM into all windows containing the name netparcel Sends keystrokes AlTwThTM into all other windows

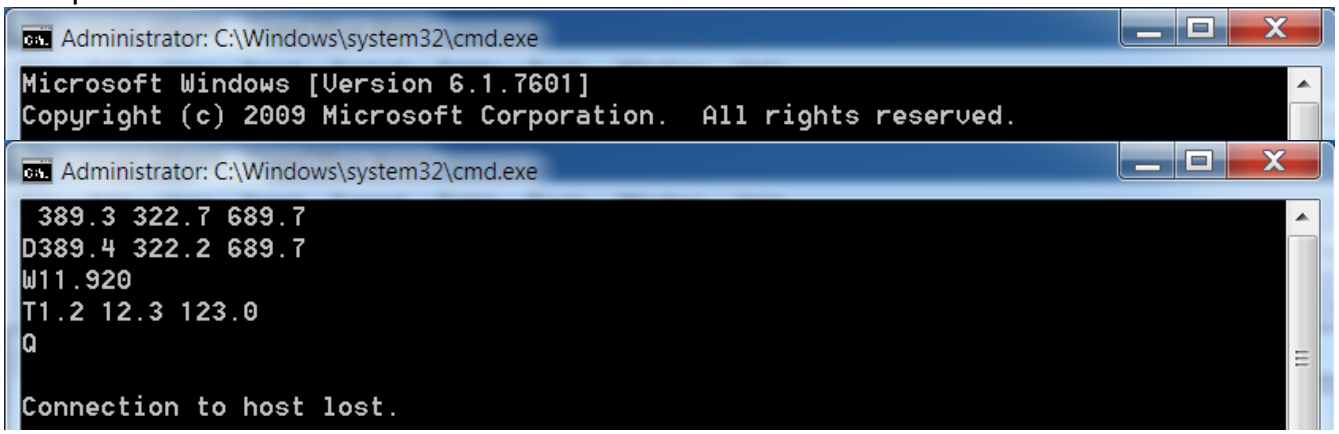
Communicate Using TCP/IP

It is recommended to use TCP/IP integration method if you are using a Windows or Linux Desktop application to communicate with the Tricolops Dimensioner.

1. The application will listen to TCP port ***59086*** on ***localhost (127.0.0.1)*** while it is running.
2. Once TCP communication is established, the program will respond to the following ASCII character commands:

Command	Function	Example reply
D	Dimensions of the object in millimeters (dimensions are always in mm)	0.0 0.0 0.0 (if no object detected / camera not connected) 12.3 13.3 14.3 (object detected)
d	Dimensions of the object (as displayed)	0.0 0.0 0.0 (if no object detected / camera not connected) 12.3 13.3 14.3 (object detected)
W	Weight captured by the scale in kg (weight is always in kg)	0.41
w	Weight of the object (as displayed)	0.41
V	Application version	1.5.0
S	Serial number of the dimensioner	17122430000
s	Dimensioner status	Place objects in trigger area
T	Send test dimensions (Can be used without a dimensioner connected)	1.23 12.3 123.0
Q	Terminate communication (the communication socket will be closed)	N/A
K	Terminate Tricolops Dimensioning App (the communication socket will be closed and Tricolops Dimensioning App will exit)	N/A

Sample Screenshot



Communicate Using RESTful API

It is recommended to use the RESTful API integration method if you are using a web application to communicate with the Tricolops Dimensioner

1. The application will listen to HTTP port *8080* on *localhost (127.0.0.1)* while it is running.
    - The port can be configured by going to
      Advanced – Settings – Advanced – Webserver Port
2. The API end point is **http://127.0.0.1:8080/data**, assuming the webserver port is not changed. For Chrome, Microsoft Edge, and other Chromium based browers, the flag *chrome://flags/#block-insecure-private-network-requests* needs to be disabled.
3. Sample data
   NOTE: dimensions are always in mm, weight is always in kilograms


```
{
  "length": "388.113037",
  "width": "321.346466",
  "height": "689.845215",
  "weight": "11.920000",
  "status": "7",
  "message": "Object detected"
}
{
  "length": "0.000000",
  "width": "0.000000",
  "height": "0.000000",
  "weight": "0.000000",
  "status": "0",
  "message": "Device not connected"
}
```


4. Status codes

Status	State
0	INIT – Application is running, but no dimensioner is connected
1	CONNECTING – Application is attempting to connect to a dimensioner
2	REQUIRE_SETUP – Additional setup is required (trigger area needs to be set)
3	CONNECTED – Dimensioner is connected successfully
4	WARMUP – Dimensioner is warming up
5	NOT_READY – Trigger area is not flat / not perpendicular to camera
6	READY – Dimensioner is operational but no packages are detected in the trigger area
7	DIMENSIONING – A package is detected in the trigger area, if the package is measured successfully, length, width, and height will be non-zero. Check messages for more details regarding the measurement status.

5. Messages
   Messages are important when status is 7 – DIMENSIONING, the list of possible messages and their interpretations are:

Message	**Explanation**
Object is not fully in view	The package is not fully in the view of the camera, possible scenarios are: • The operator is scanning a barcode on the box and their hand / barcode scanner is in the way. If "Keep History" in Advanced -> Settings is turned on, the measurements will be the last successful measurement of the box, otherwise, the measurements will be all zeros. • The package needs to be re-positioned to be fully in the view of the camera, the measurements will be all zeros. • The package is too large to be measured, the measurements will be all zeros.
Object measurement is not stable	The measurements are not stable, possible scenarios are: • The package is shiny or has lots of fine features • The package is still moving The measurements may not be accurate.
Object is not a cuboid	The object is not a cuboid. If "Display Non-cuboid Parcel" in Advanced -> Settings is turned on, and the package failed cuboid test parameters, this message will be displayed. The measurements will be the smallest hexahedron within which the object can be contained.
Object is oversized	If "Display Oversized Parcel" in Advanced -> Settings is turned on, and the package larger than the maximum dimensions, this message will be displayed. The measurements will be the object's dimensions.
Object cannot be measured	The object cannot be measured for other reasons. The measurements will be all zeros.
Object measured successfully	The measurements will be the object's dimensions.

6. Sample JavaScript code
   With the Tricolops Dimensioning Application running, sample JavaScript code can be found by going to http://127.0.0.1:8080 (port can be customized in Advanced – Settings – Advanced – Webserver port). Once webpage is displayed, right click on the web page and select View Source or View Page Source. Contact support@tricolopstechnology.ca to request a standalone HTML page with sample JavaScript code to communicate with the dimensioning application.

   The same code can be found on https://content.tricolopstechnology.ca/resources/home.html, navigating to this URL will display camera images and dimensions when the Tricolops Dimensioning Application

is running on the target computer. For Chrome, Microsoft Edge, and other Chromium based browsers, *chrome://flags/#block-insecure-private-network-requests* needs to be disabled.

Acquiring Images Using RESTful API

With the Tricolops app running, you can access the JPEG images captured by the dimensioner through issuing GET request to http://127.0.0.1:8080/video.

High definition video images can be captured by dimensioners shipped after January 1st, 2019. To enable high definition video image, go to Advanced → Settings → Advanced, and enable “HD Video”.

If you would like the images labelled, the following URL will return images labelled according to parameters:

URL Parameter	**Result**
box=true	The outline of the parcel will be drawn on the image
dimension=true	The dimensions of the parcel will be printed on the image
weight=true	The weight of the parcel will be printed on the image
area=true	The trigger area will be outlined on the image

For example:
http://127.0.0.1:8080/video?box=true&dimension=true&weight=true will return an image with the outline of the box, as well as dimensions and weight of the box printed at the bottom of the image.

Acquiring History Data Using RESTful API

With the Tricolops app running, you can access the history and images captured by the Tricolops History application through issuing POST request to http://127.0.0.1:8080/history.

Input Parameters

Parameter	Description
start	**Integer,** Number of milliseconds since Epoch. If this parameter is specified and greater than 0, only entries saved at or after "*start"* time will be returned.
end	**Integer,** Number of milliseconds since Epoch. If this parameter is specified and greater than 0, only entries saved at or before "*end*" time will be returned.
offset	**Integer,** If this parameter is specified and greater than 0, the first "*offset*" rows are excluded from the returned result.
count	**Integer,** If this parameter is specified and greater than 0, the number of rows returned is limited to "*count*" rows, otherwise number of rows returned is limited to 250 rows.
barcode	**String,** If this parameter is specified and not empty, only entries with barcode matching "*barcode*" is returned.
exact_barcode	**Boolean,** If this parameter is specified and set to True, only entries with barcode that match "*barcode*" exactly is returned, otherwise all entries with barcodes that is like "*barcode*" will be returned.
quantity	**Integer,** If this parameter is specified and greater than 0, only entries with quantity equal to "*quantity*" will be returned.
company	**String,** If this parameter is specified and not empty, only entries with company matching "*company*" will be returned.

Output Parameters

The returned results are arrays of objects that represents each row in the Tricolops History table. The parameters of each object represent the column names, they are:

Parameter	Description
barcode	**String,** the barcode associated with the entry.
timestamp	**Integer,** number of milliseconds since Epoch that the entry is captured.
company	**String,** the company associated with the entry.
quantity	**Integer,** the quantity associated with the entry.
weight	**Integer**, the weight associated with the entry.
length	**Integer**, the length associated with the entry.
width	**Integer**, the width associated with the entry.
height	**Integer**, the height associated with the entry.
video_uri	**String,** the relative path for the captured image. This path is dynamically generated every time a POST request to /history is issued. Once a POST request to /history is issued, all past video_uri are invalidated.

Example Input

Empty JSON object

```
{
}
```

Example Output

Up to 250 entries are returned.


```
[{
  "barcode": "fc049b4d-fc84-4457-92a4-fdf5f76f8b86",
  "company": "",
  "height": 10.199999962002039,
  "length": 27.399999897927046,
  "quantity": 1,
  "timestamp": 1559243584285,
  "video_uri": "/history/video/0",
  "weight": 0.0,
  "width": 25.399999905377626
}, {
  "barcode": "b5bb1e84-32e6-4a0c-9556-b3ffbb16d9fd",
  "company": "",
  "height": 10.799999959766865,
  "length": 27.199999898672104,
  "quantity": 1,
  "timestamp": 1559243580204,
  "video_uri": "/history/video/1",
  "weight": 0.0,
  "width": 25.399999905377626
}]
```


Example Input 2

Return all entries after 5/30/2019, 3:13:00 PM GMT, skip the first 200 entries and limit the number of returned entries to 200.

```
{
"start": 1559243580204,
"offset": 200,
"count": 200
}
```

Example Input 3

Return entry that matches the specified barcode exactly.

```
{
"barcode": "b5bb1e84-32e6-4a0c-9556-b3ffbb16d9fd",
"exact_barcode": true,
"count": 1
}
```

Example Input 4

Return entries that matches parts of the specified barcode.

```
{
"barcode": "b5bb1e84-"
```

```
}
```