Getting Started with CloudLink

v8.0.1, October 2016

Disclaimer

The information and specifications described in this manual are subject to change without notice.

——————Contents

Preface

Welcome

Purpose of This Guide

This guide describes how to use di-soric's CloudLink to visualize di-soric Link values and images from the CS50 Sensor. CloudLink runs on the web browser of your choice, including those found on tablets and smart phones.

Preface

1

CHAPTER 1 — CloudLink Overview

This section provides a general description of the CloudLink interface.

Introduction

CloudLink allows you to visualize **di-soric Link** values and images from the CS50 Sensor. CloudLink runs on the web browser of your choice, including those found on tablets and smart phones.

Requirements

CloudLink requires an HTML5-compatible browser:

- Internet Explorer 10 or later

- Google Chrome

- Firefox

- Mobile Safari (iPhone / iPad)

- Mobile Chrome on Android devices

The following browswers were explicitly tested for compatibility:

- Internet Explorer 10.0.9

- Internet Explorer 11.0.2

- Google Chrome 33.0

- Firefox 28.0

Additional Notes:

- Windows Safari is not supported.

- Internet Explorer 11 or later and Google Chrome 33 or later are recommended for extended CloudLink sessions.
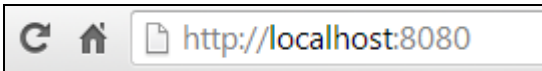
Connecting

To launch CloudLink, use your favorite web browser and enter the address of your device in the browser's address bar. For example, if you have a CS50 Sensor on your network at address **10.20.1.123**, you would enter:

C ⌂ http://10.20.1.123
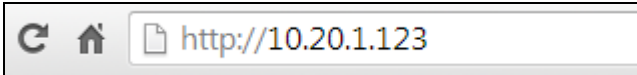
CloudLink also works with **CS50 Software**'s Emulator.

To connect to a software-based job running CS50 Software:

First, be sure the job is running, and then type the following into your browser's address bar:

C ⌂ http://localhost:8080

Note: You must specify port **8080** for a PC-based connection. If you are connecting to a PC-based system from a different machine on the network, use the IP address of the PC instead of the local host. For example, use **http://10.20.1.234:8080** if the PC's IP address is **10.20.1.234**.

Once you press the **Enter** key, you should see the following home page:

Click the **CloudLink Dashboard** icon to launch the application.
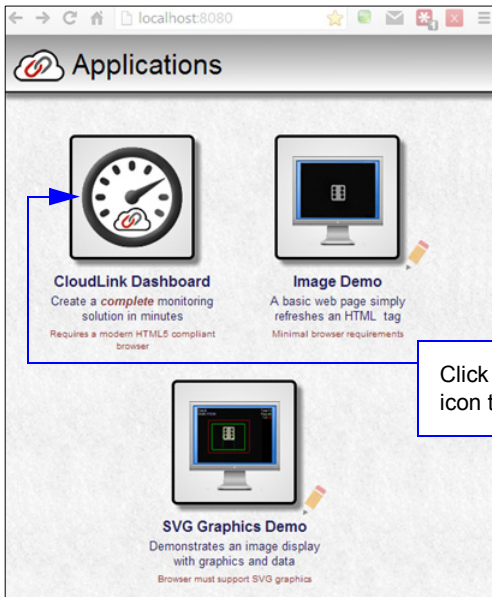
Application Overview

The CloudLink Dashboard user interface is a single page web app-style application. Most web pages show information that can typically extend beyond the bottom of the browser window, requiring the user to scroll to see it. They typically contain links to other pages, which is how the user navigates around a web site.

In contrast, CloudLink behaves more like an application, expanding to fill the browser window, and automatically adapting to any changes in the dimensions of the window. Although CloudLink allows you to define and use multiple display pages, they are all contained within a single web page.

The browser stores a maximum of 50 images. The sensor stores images, image thumbnails, and data records that can be requested via the web page or API. When memory is full, the sensor will first delete full-size images, then thumbnails, and ultimately the data records on a first in-first out basis.

This is a typical view of a CloudLink page.
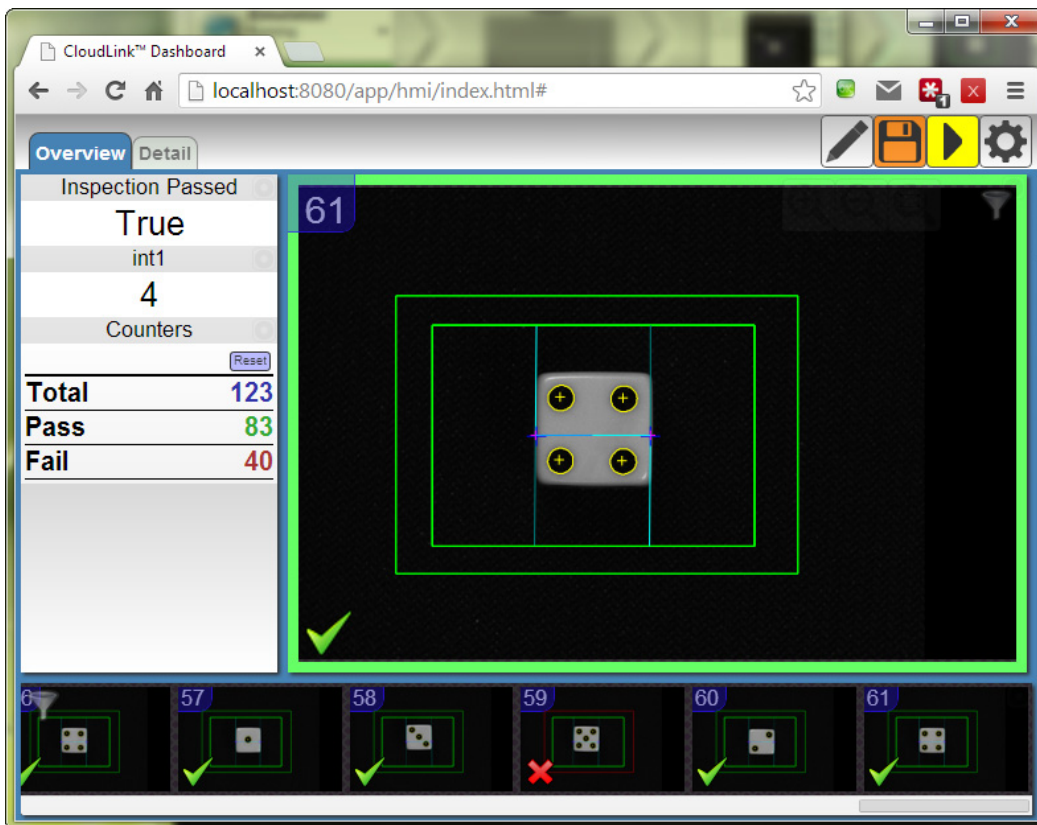
Note: CloudLink does not currently support display of color images from color cameras.

Application Bar

The **Application Bar** is located at the top of the CloudLink interface.

| Overview Detail | | ✏️ 💾 ▶️ ⚙️ |

There are three components to the Appplication Bar:

- **A set of page selection buttons.** In the example above they appear as tabs, but the style can be customized if necessary.

- **A logo.** The position and contents of the logo can be customized.

- **A toolbar.** The toolbar provides access to various CloudLink settings and modes. The position and size of the toolbar can be customized.

Pages, Panels, and Widgets

The main area of CloudLink displays one of a number of **pages**. If there is more than one page defined, you can switch between them by using the page selection buttons on the application bar, or by using the arrow keys on your keyboard.

Each page is organized into a set of regions called **panels**. The following image shows an empty page to demonstrate the arrangement of the panels on a page.
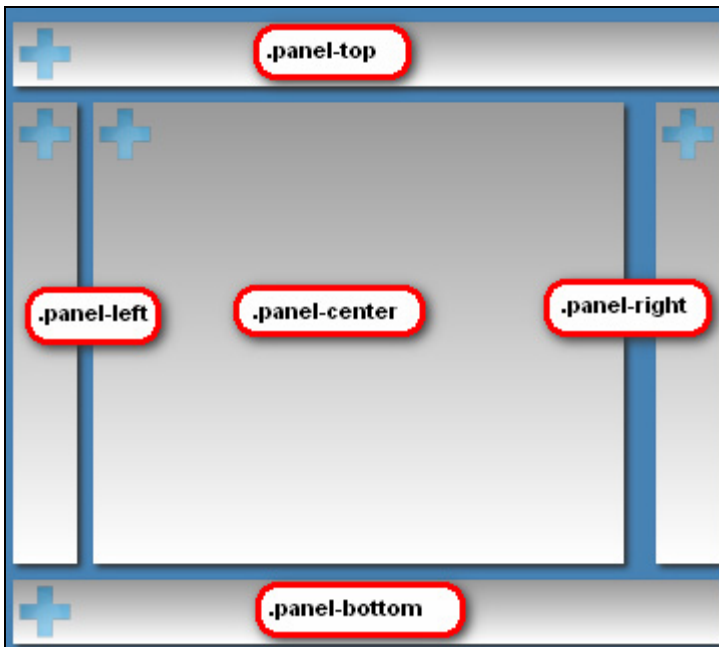
Note the names, which start with **.panel-**. It is not necessary to know these names to use CloudLink; however they do have significance if there should be need for customization. Customizing CloudLink requires the modification of CSS (Cascading Style Sheets). The names shown correspond to the CSS class selector for that panel.

The purpose of the panels is to act as containers for a number of **widgets**. Each widget has the ability to visualize and interact with one or more items of inspection data such as di-soric Link values, inspection counters, timing information, or images.

Each panel has special layout and behavior properties that can be exploited to create a wide variety of different layouts. The following table summarizes the position and properties of each panel:

Panel	Position	Properties
.panel-top .panel-bottom	Docked at the top or bottom of the page, fully stretching from the left edge to the right edge of the window.	Certain types of widgets are automatically stretched to fill the panel space horizontally. For example, if you should add a chart or a filmstrip to these panels, by default they will automatically stretch.
.panel-left .panel-right	Docked at the left or right of the page, they extend between .panel-top and .panel-bottom. The width of these panels is fixed (by default 194 pixels)	Widgets in these panels are typically stretched to fit exactly in the fixed width. If there is not enough vertical room to display all the content, scroll bars are made available.
.panel-center	This panel automatically stretches to fill the center area bounded by the other panels	An image widget placed in this panel automatically stretches to occupy the entire panel area. Any other widgets would then appear over the image.

If a panel does not have content (i.e. no widgets are placed in it), it is hidden from view, with the other panels adjusted to occupy the available space. Examples of possible page layouts:
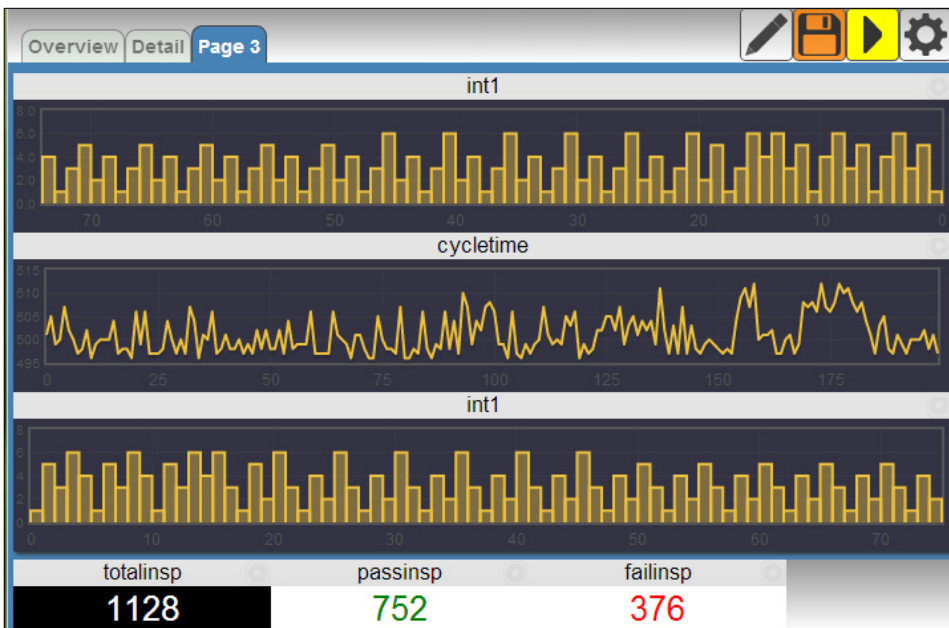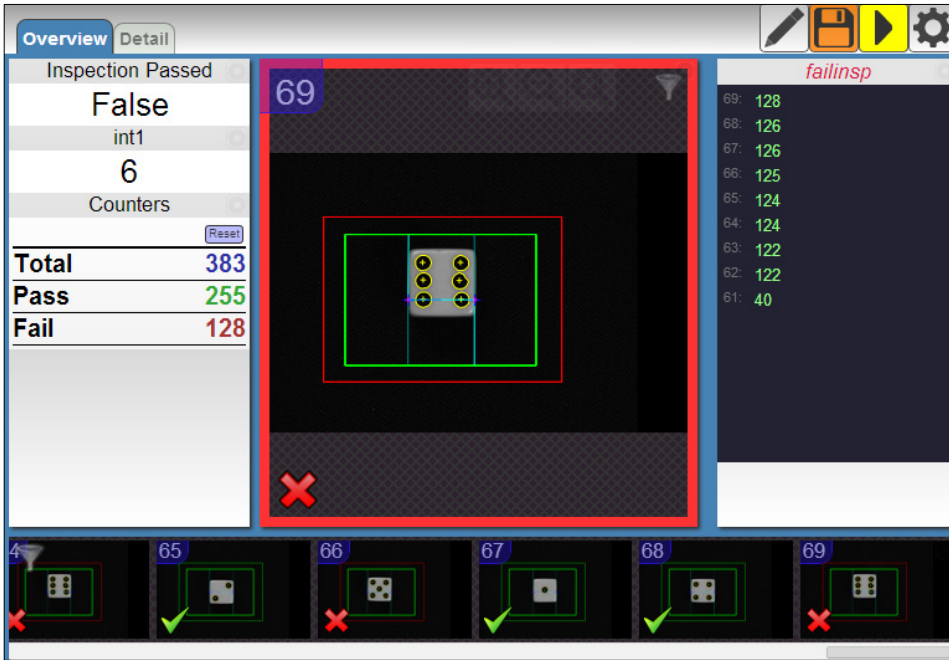
Data Event Identifiers

At the end of each inspection cycle, the relevant result data is collected into a record. This data is coherent, meaning that all the values are part of the same result, and therefore represent a snapshot of information as it was at that particular point in time. When CloudLink fetches this data, it associates each of the data records with a unique **Data Event Identifier**. This identifier is visible in various CloudLink widgets, and can help when making correlations between various pieces of data. It is also used internally, for example to highlight all items related to the same event when hovering over a displayed value or image.

Live and Review Modes

CloudLink has the following two fundamental modes of operation:

- **Live Mode:** In this mode, CloudLink is continuously fetching new data from the connected device. The amount and frequency of data received is balanced by bandwidth and performance constraints.

- **Review Mode:** In this mode, CloudLink is not receiving new data. Instead, you can view and analyze already captured historical data.

To switch between **Live Mode** and **Review Mode**, use the appropriate toolbar button in the **Application Bar**, or press **CTRL-R** on the keyboard.

When in review mode, clicking on one of the items in a historical widget (for example, a filmstrip, log, or chart) will automatically load all the other widgets with data with the same Data Event Identifier. Clicking on **Filmstrip Entry** will load the image into the image widget, and display the inspection counts, di-soric Link values, and other elements as they were at that time. Likewise, you can click on a log entry with similar results.

CloudLink Web HMI Storage Limits

Up to the last 100 records are stored, including at least one failure record. If the last failure record is older than the most recent 100 records, that record plus the latest 99 passing records are stored for a total of 100 records.

Up to 100 of the latest SVG records are stored on the PC-based system.

The number of images stored on the system is dependent on the sensor configuration's buffer count. At most this will be the buffer count as defined in the system's Camera Definition and Buffer Counts setting. This count is reduced by buffers being processed or waiting for triggers in a pipeline mode. For example, a sensor with 16 buffers in pipeline triggering mode with one image in process would have a maximum of 14 of the most recent image records saved for retrieval.

Chapter **1** CloudLink Overview

2

CHAPTER 2 Layout Editing

This section describes how to optimize the layout of the CloudLink interface for your application.
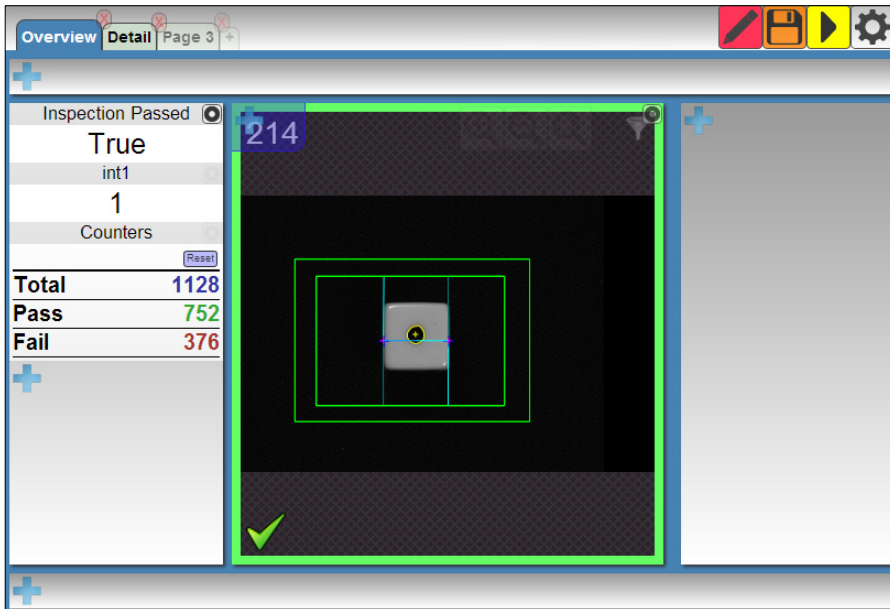
Layout Edit Mode

Layout Edit Mode can be toggled at any time by pressing this toolbar button:

or by pressing **F10** on the keyboard.

In Layout Edit Mode, all panels are visible even if they are empty of widgets:

Notice that the top, right, and bottom panels can now be seen even though they are empty of widgets.

When a change is made to the layout, the **Save** icon in the toolbar will be highlighted.

Before Change:

After Change:

Clicking the **Save** button will save the layout changes.

Creating, Editing, and Removing Pages

When in **Layout Edit Mode**, an extra button appears to the right of any existing page selector buttons. Pressing this button will create a new empty page.

To change the title of a page, double-click the title (or long-press on a touchscreen device). Type a new title and press **Enter** to accept or **Esc** to cancel.
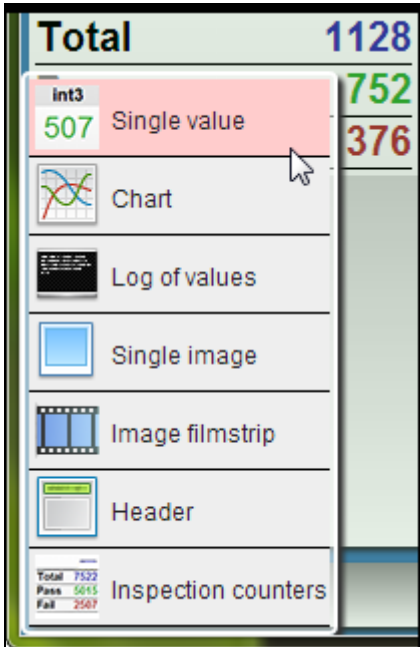

To delete a page, click on the **X** that appears in the upper right corner of the page selector button. Note that this is only available when in Layout Edit Mode.
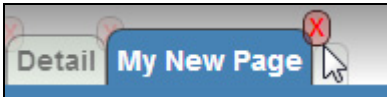
Inserting and Configuring a Widget

When in Layout Edit Mode, note the **+** buttons in each panel. Pressing one of these buttons will bring up a menu of available widgets to insert.
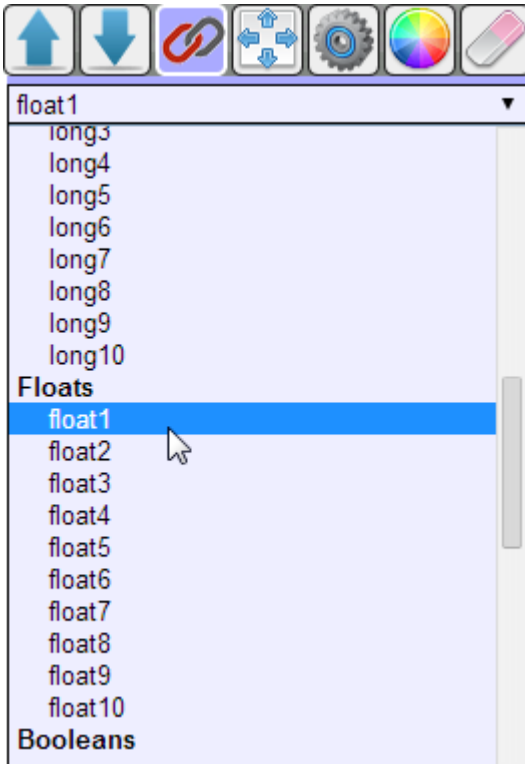

Clicking **Single Value**, for example, will insert a widget that displays one di-soric Link value. The settings panel for the newly inserted widget is shown automatically:
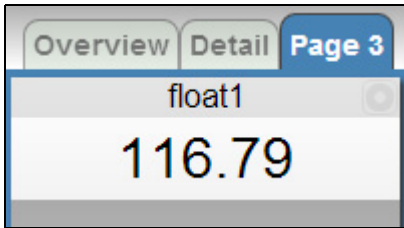
Buttons on the settings panel may vary from widget to widget. Clicking on the di-soric Link icon:
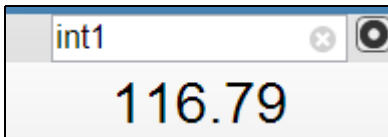

will show a combo box selector allowing a choice of di-soric Link tags to monitor in the widget.
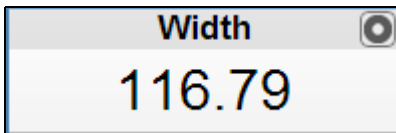
Selecting **float1** in this example will "link" the widget to the di-soric Link tag float1:

To change the title of the widget, double-click the header (or long-press on a touchscreen device).

Type a new title and press **Enter** to accept or **Esc** to cancel.

The Widget Settings Panel

Notice the ⊙ button in the upper right corner of the widget. This button appears on all widgets (but is nearly invisible unless you hover over the widget). Pressing the button will show the settings panel for that widget.

The buttons you will see on the settings panel include the following:

Icon	Function	Supported Widgets
⬆	Move widget up within its panel. Will be disabled if widget is the first one in the panel.	All
⬇	Move widget down within its panel. Will be disabled if widget is the last one in the panel.	All
🔗	Link the widget to a Link tag. Pressing this button shows a selector with tag names organized into categories.	Single Value, Chart, Log
⚙	Change settings specific to the widget. For example, the Single Value Widget settings allow specification of dimensional units or to change the text displayed when a boolean is true or false.	Single Value
✛	Change the size of the widget. Some common choices are: min — Fit into the minimum possible width. Widget will stretch horizontally if necessary to fit content fixed — The width of the widget will be fixed to the width of the left and right panels. This width can be customized, but by default it is 194 pixels. half — The width of the widget will be fixed to half of the width of the left and right panels. This allows two widgets to sit side by side within a side panel. stretch — The width of the widget will be extended to fill the width of the panel.	All – choices depend on widget type
🎨	Change the text color of the widget to one of a fixed pallet of choices.	Single Value
👁	Change the view mode of the widget. Certain widgets will have different options for how to display information. Press this button to select one of the viewing options.	Chart
✏	Delete the widget	All

Chapter **2** Layout Editing

3

CHAPTER 3 Tags

This section provides a list of tags and describes their functions.

CloudLink Tags

CloudLink widgets can be "linked" or associated with a number of data sources. These mostly correspond to di-soric Link tags, but CloudLink offers additional items that are not strictly di-soric Link tags. This document will refer to all of these items as tags. The tags available are:

Category	Tag Name	Function
Status	inspPassed	The pass/fail status of the inspection
Counters	totalinsp	A count of the total number of inspections
	passinsp	A count of the number of passed inspections
	failinsp	A count of the number of failed inspections
	totalinsp-interval	A special version of totalinsp which can be reset to zero at any point in time. Resetting this count does not affect the actual count maintained by the vision system.
	passinsp-interval	A special version of passinsp which can be reset to zero at any point in time. Resetting this count does not affect the actual count maintained by the vision system.
	failinsp-interval	A special version of failinsp which can be reset to zero at any point in time. Resetting this count does not affect the actual count maintained by the vision system.
Statistics	cycletime	The processing of the inspection cycle in milliseconds
	idletime	The time not spent processing between one inspection start to the next. Measured in milliseconds.
	drawtime	The time spent by the inspection drawing graphics. Measured in milliseconds.
Integers	int1 – int10	The first 10 Link integers
Longs	long1 – long10	The first 10 Link longs
Floats	float1 – float10	The first 10 Link floats
Booleans	bool1 – bool10	The first 10 Link booleans
Strings	string1 – string10	The first 10 Link strings
Internal	log	A log of diagnostic information
	recordId	The internal record identifier associated with the current item

4

Widgets

This section gives a detailed description of each widget available in the CloudLink interface.

Single Data Value Widget

The **Single Data Value** widget shows one value and a title.
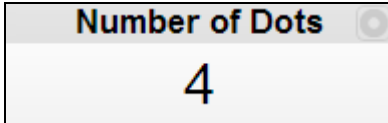
The title can be edited by double-clicking on it (or long-pressing on a touchscreen device). To link the displayed value to a tag, use the settings button:

and select the di-soric Link tag selector:

from the settings panel. Select the color selector:

to select a different text color for the displayed value.

4

Widgets

With the **Linked Status** option set to **no link**, you can use the **Text Color** and **Background Color** color options to change the colors of the **Single Value Widget**.
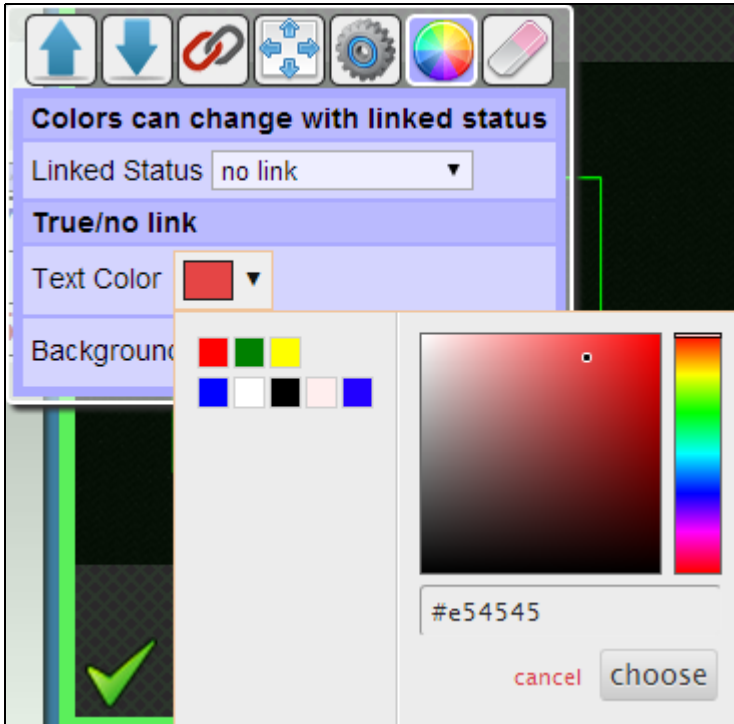
The **color picker** allows you to select colors in several different ways. You can click on the spectrum or specify a hex color value. Previous color choices are also available as a palette.
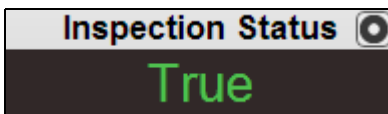
It is also possible to change the colors based on the true / false value of an additional linked status item. For example, you can change the look of a displayed measurement based on whether the inspection has passed or failed. Selecting **inspPassed** for a **Linked Status** will show additional color options.

Colors can change with linked status

Linked Status   inspPassed

no link
Status
inspPassed
Booleans
bool1
bool2
bool3
bool4
bool5
bool6
bool7
bool8
bool9
bool10

True/no link

Text Color

Background C

Colors can change with linked status

Linked Status   inspPassed

True/no link

Text Color

Background Color

False

Failed Text Color

Failed Background Color
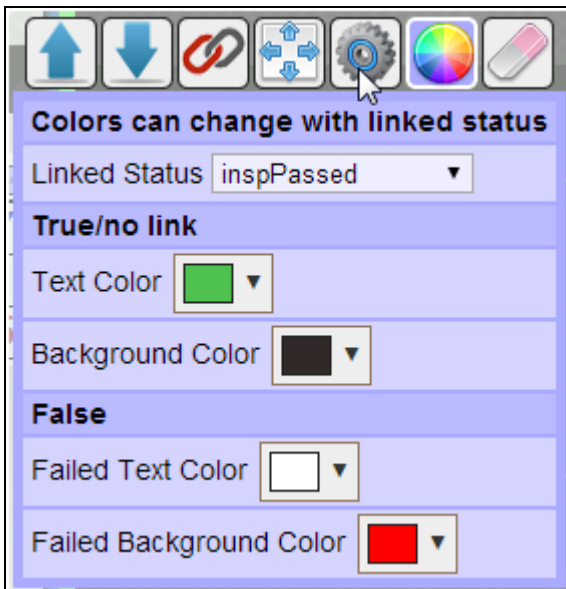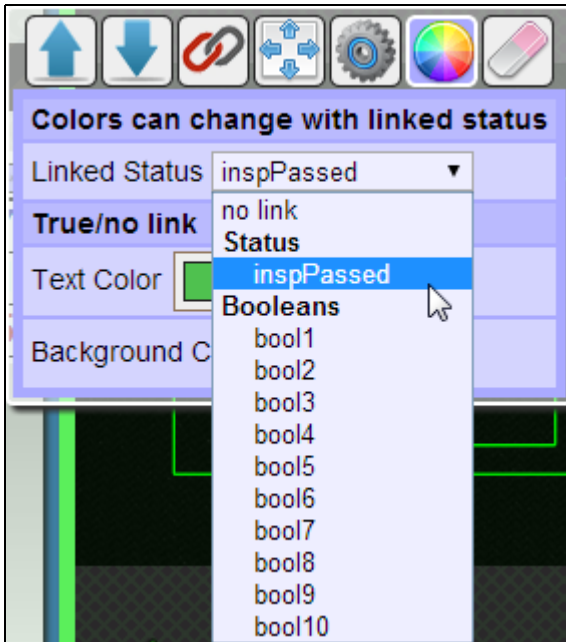
The specified colors will now be used depending on the value of **inspPassed**.
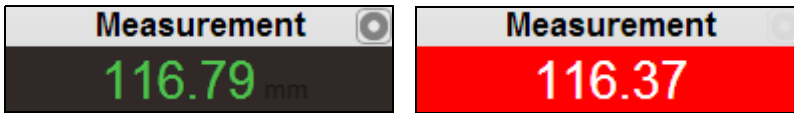
Selecting the widget options button:

will display additional setup options.
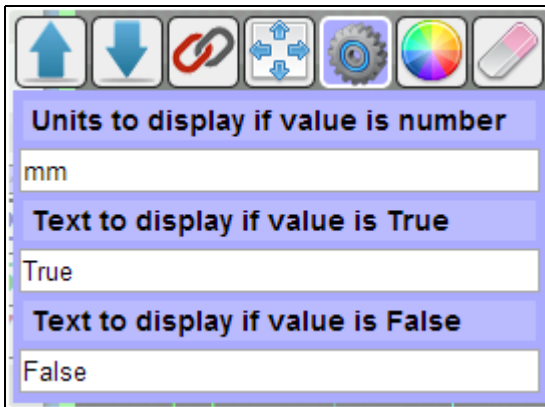
If desired, text representing the units to display can be specified. This will only apply if the value displayed is a number.
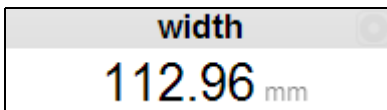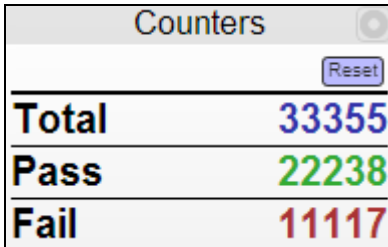
If the value displayed is boolean, the text to display for True / False can be changed. For example you may want to change this to Passed / Failed.
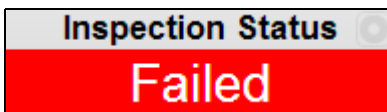
Counters Widget

The **Counters** widget displays the values of the interval version of the inspection counts (**totalinsp-interval**, **passinsp-interval**, **failinsp-interval**). This allows the counters to be reset to zero without affecting the actual counts on the vision system.

The title can be edited by double-clicking on it (or long-pressing on a touchscreen device).

Log Widget

The **Log** widget shows the values of the linked tag as a historical list. The most recent value is at the top, and older values are listed in descending order below.
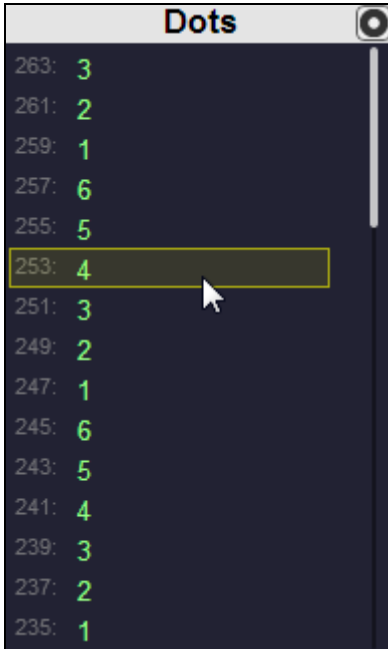
To the left of each displayed value is a number representing the corresponding Data Event Identifier. This will correspond to similar values seen in other widgets. Hovering over one of the values higlights all values that correspond to the same event. For example, if a filmstrip is displayed, the image that corresponds to the highlighted value will also be highlighted.

The title can be edited by double-clicking on it (or long-pressing on a touchscreen device). To link the log values to a tag, use the settings button:

and select the di-soric Link tag selector:

from the settings panel.

Chart Widget

The **Chart** widget shows a chart of tag values over time. There are two types available: **Bar** and **Line**.

The chart is automatically scaled so that the minimum and maximum values are contained on the chart, and the amount of data shown corresponds to the available history depth.

Data on the chart widget reads from right to left, with the most recent values on the right. When a data event is hovered over, the corresponding chart value is highlighted.

The chart title can be edited by double-clicking on it (or long-pressing on a touchscreen device). To link the chart to a tag, use the settings button:

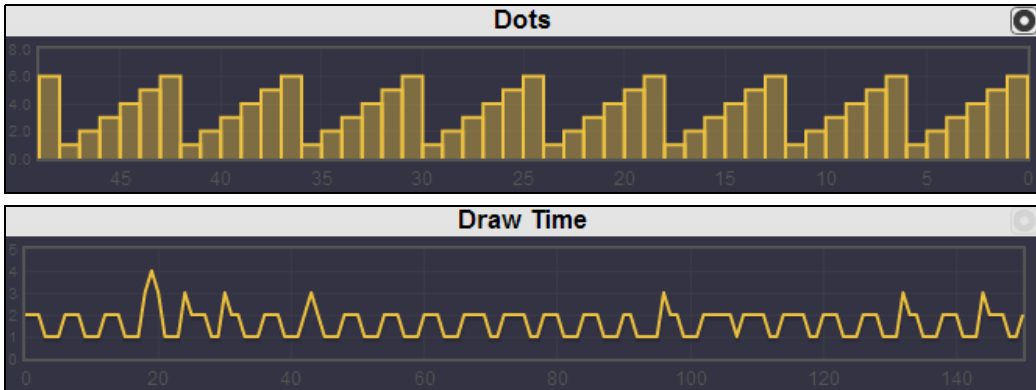and select the di-soric Link tag selector:

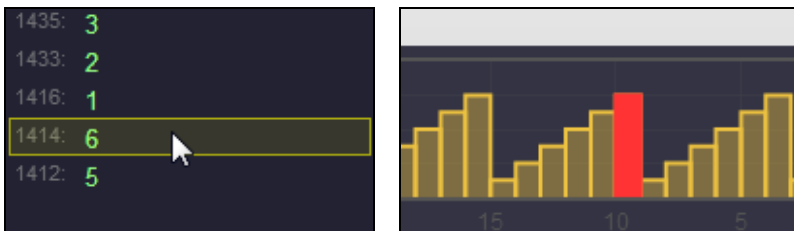from the settings panel. To select a different chart type, select the **View Mode** button:

from the settings panel.

Image Widget

The **Image** widget represents the most central function of CloudLink. It can show an image and its associated graphics.

Note: CloudLink does not currently support display of color images from color cameras.

The image widget also permits zooming and panning, supporting pinch gestures on touchscreen devices. The graphics are rendered as vector graphics, so zooming in shows more detail instead of making the lines thicker.

Note: There is currently a known Internet Explorer issue with displaying scalable zoomed-in vector graphics. When using Internet Explorer, the lines will appear thicker as you zoom in.

When added to the center panel, the image widget automatically expands to fully occupy the area. Any other widgets added will appear over the image.

The image widget also displays a thick colored border representing the inspection status (**green = pass, red = fail**). In the bottom left corner either a **green check** or **red X** is displayed, corresponding to the inspection status indicators within the filmstrip widget.

The **Data Event Identifier** is displayed in the upper left corner.

A zoom toolbar appears when hovered over.
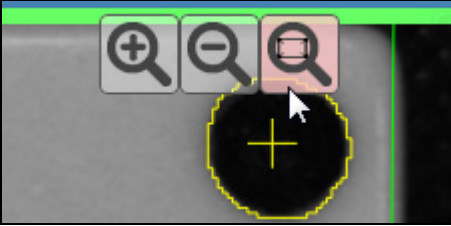
Click ⊕ to zoom in and ⊖ to zoom out. The ⬚ button zooms and centers the image to be contained completely within the widget. If using a mouse, the scroll wheel can be used to zoom in and out. If using a touchscreen device, pinch gestures can be used.

Filmstrip Widget

The **Filmstrip** widget shows a history of images. The images are displayed from right to left, with the most recent images appearing on the right side.
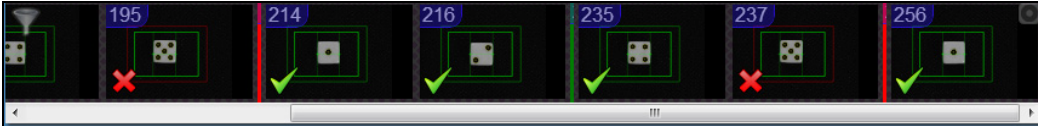
Each filmstrip entry shows the inspection status by displaying either a **green check** or **red X** in the lower right corner. The data event ID is shown in the upper left corner.

It is important to note that not every image can be captured by CloudLink. The number of images available depends on bandwidth and timing considerations. Therefore, it is likely that some images will be missing (even though the rest of the data for that inspection is available). Any missing images are represented in the filmstrip by vertical bars colored **red** for failed inspections or **green** for passed inspections.

Hovering over an item in the filmstrip will highlight all data corresponding to the same data event. You can also hover over the missing image bars.
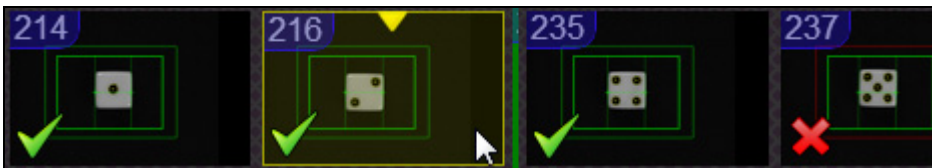
Clicking the filmstrip entry will select that item for review. This will load the widgets with the data as it was at the time the event was recorded. If the image is missing, any image widgets will display a **missing image** placeholder.
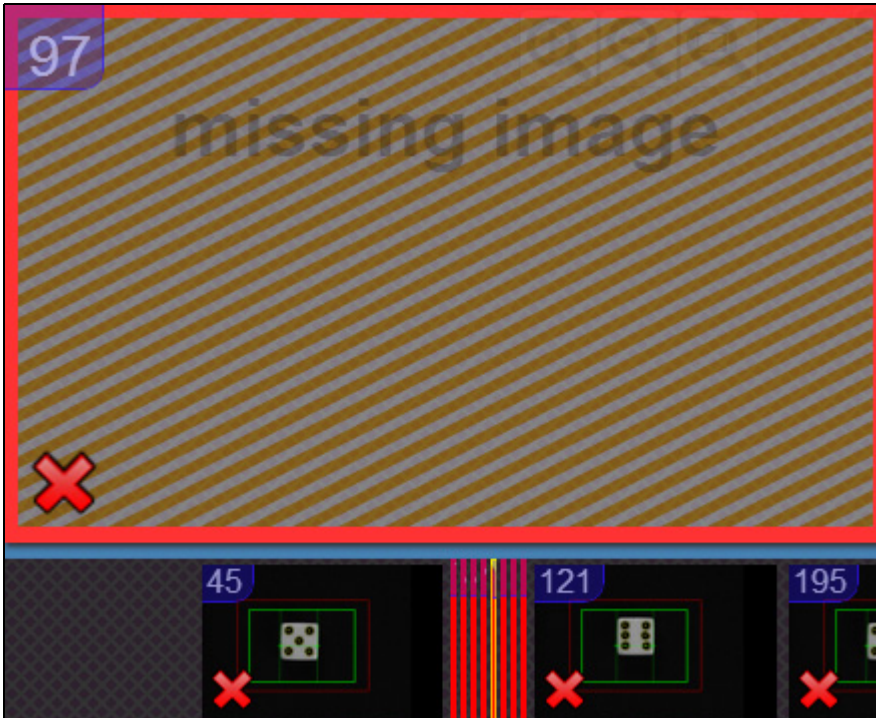
In the upper left corner of the filmstrip widget is a **filter button**. Clicking this button will filter what is shown in the filmstrip.
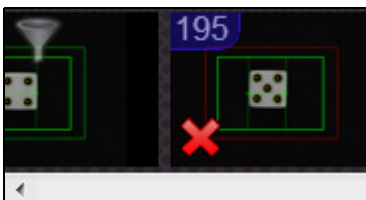
When the filter button is first clicked, only failed items will be displayed.
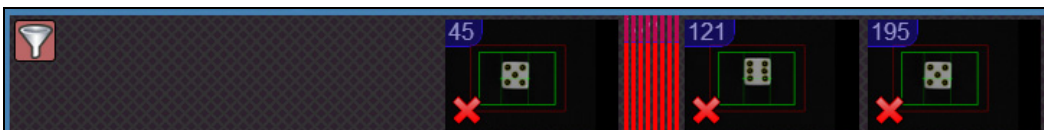
Clicking the filter button again will change the filter so that only passed items are displayed.
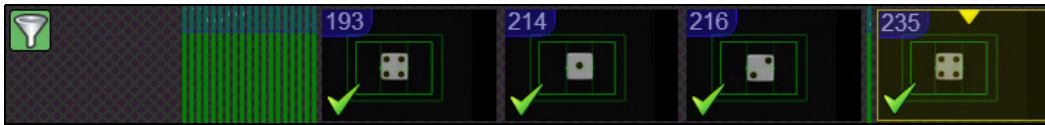
Since it is permitted to have multiple instances of widgets, you can sort pass and fail images into two groups.

Header Widget

The header widget displays a header. This can be used to identify a group of widgets if necessary.
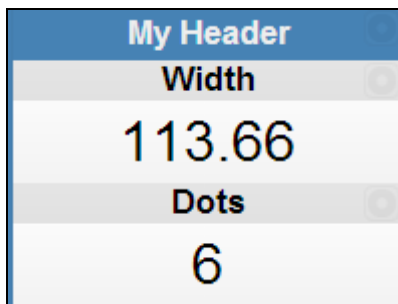
The title can be edited by double-clicking it (or long-pressing on a touchscreen device).

CHAPTER 5 REST APIs

This section describes how to use REST (Representational State Transfer) APIs to access data from the sensor.

REST APIs

REST APIs allow you to access data from the sensor over the web standard HTTP protocol.

Using a web browser, requests to the sensor are made using the IP address of the sensor as the URL root. Version 1 of the APIs are accessed using the path prefix of **/api/v1/**.

For example, to access the **SVG API** of a sensor at IP address **10.20.1.234**, the URL to the REST service would be: **http://10.20.1.234/api/v1/svg**.
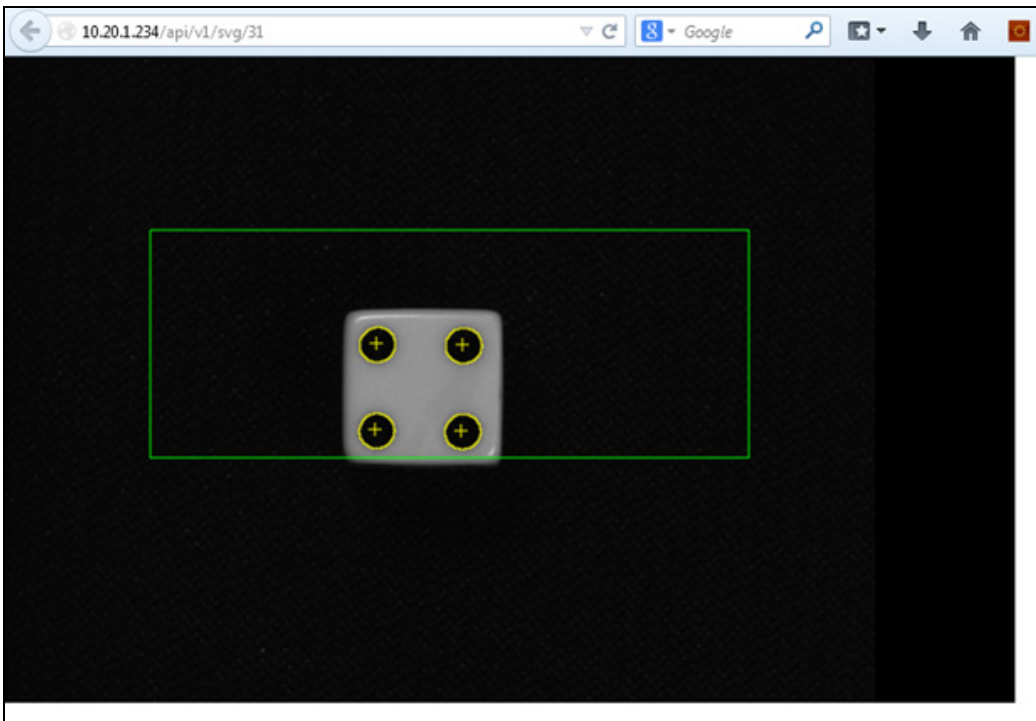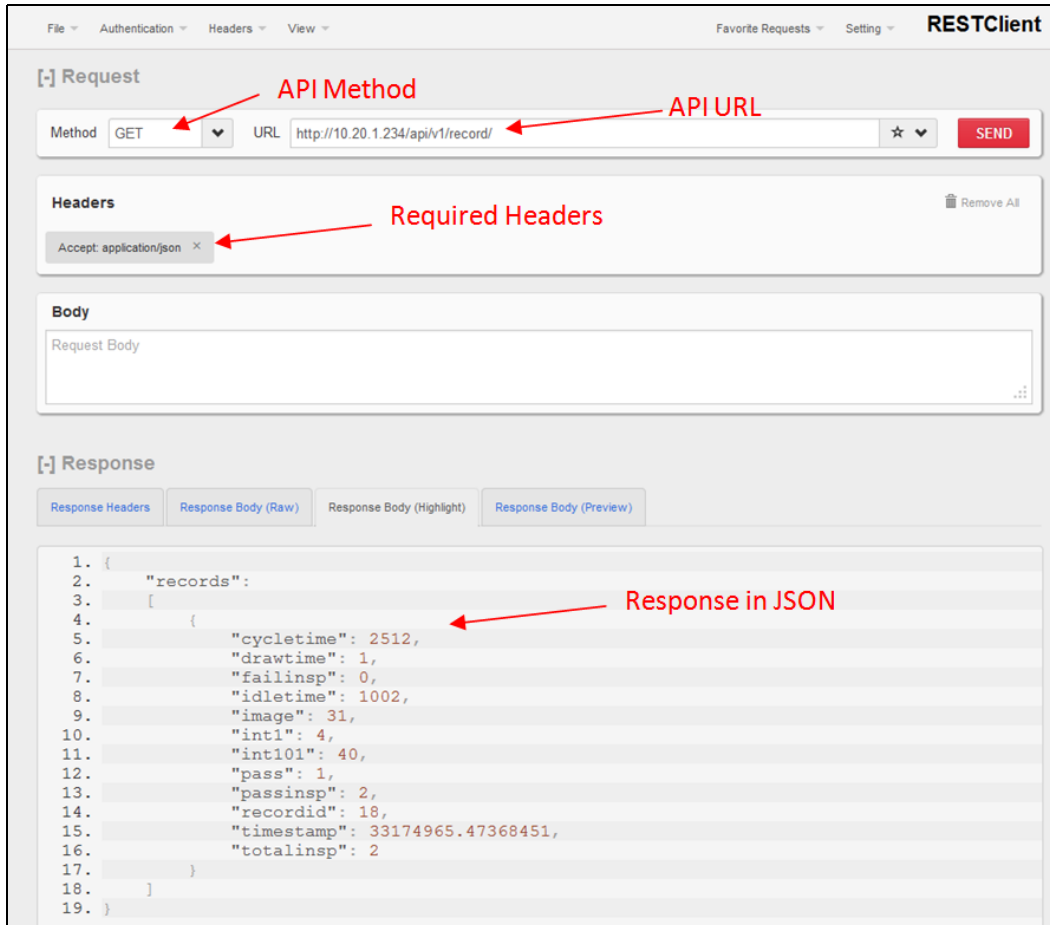
REST APIs are also used to exchange data with the sensor. To interact with the sensor's APIs using a standard web browser, a REST client browser add-on can be used. There are a variety of REST clients available for most browsers.

The following example shows **RESTClient** for the **Firefox** browser.
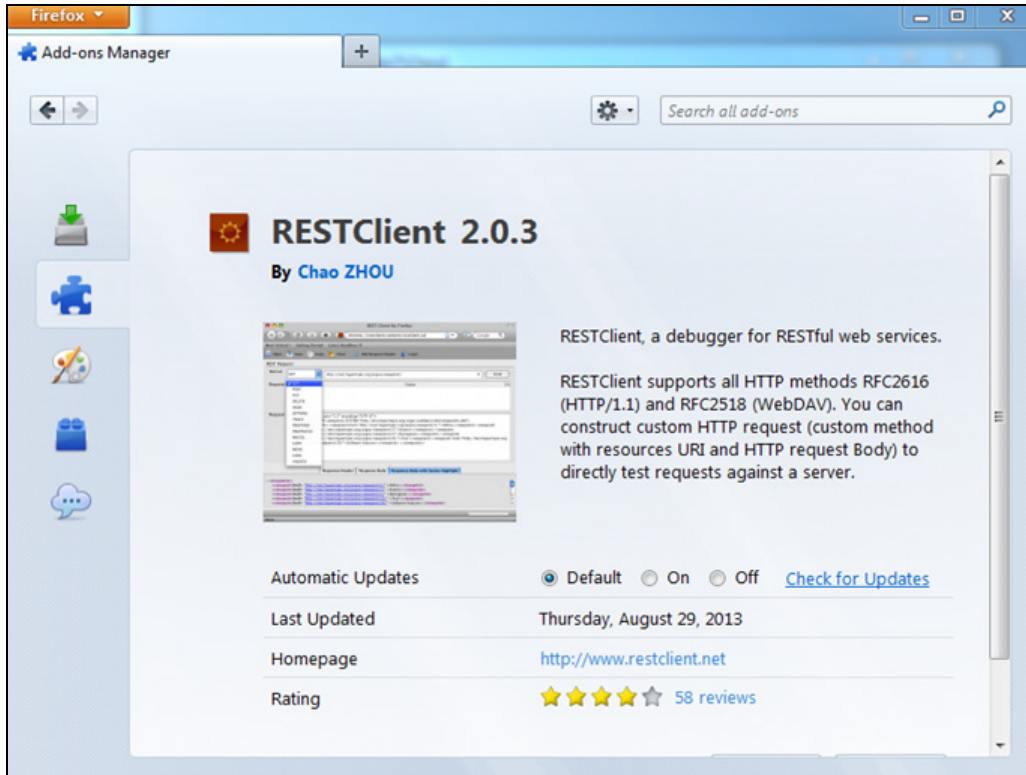
Using a REST client, a request to receive data in **JSON** format can be made to the sensor and viewed in the browser.

command

This API passes the body of the message to the serial command processor. The result string from the serial command processor is returned as the body of the response message.

The body of the message is a command string just as it would be typed into a serial terminal to the sensor, with no extra quotes or delimiters. A terminating CR or LF is not necessary.

Resource URL

```
PUT /api/v1/command

Content-Type: text/plain

Accept: text/plain

Body: "command text"
```

Response Body

```
Command response as text
```

Example

```
PUT /api/v1/command

Content-Type: text/plain

Accept: text/plain

jobinfo
```

Response Body

```
slot4=EIP_demo.avp?
```

data

This API sets or retrieves the value of a tag, tag array or all tags in a service. The response is in JSON and the header must declare it is accepted. When setting data, the body must be in JSON with a top level element named "data" with the values to be set.

Resource URL

```
GET /api/v1/data/<service>

Accept: application/json

GET /api/v1/data/<tag>

Accept: application/json

GET /api/v1/data/<tag><n>

Accept: application/json
```

parameter	value	example	description
count	\<n\>	api/v1/data/int1?count=2	number of items to return when used with \<tag\>\<n\>

```
PUT /api/v1/data/<tag><n>
Content-Type: application/json

{ "data" : "<input data>" }
```

Examples

URL

```
GET /api/v1/data/float1?count=2
```

Response Body

```
{ "data" : [1.234, 5.678] }
```

URL

```
PUT /api/v1/data/int5
```

Body

```
{ "data" : "123" }
```

stats

This API retrieves the inspection statistics for up to 10 inspections or resets the counters. The response is in JSON.

Resource URL

```
GET /api/v1/stats

Accept: application/json

PUT /api/v1/stats
```

A PUT clears the inspection counters. No other arguments are needed.

Examples

URL

```
GET /api/v1/stats
```

Response Body

```
{
"totalinsp" : [ 19, 0, 0, 0, 0, 0, 0, 0, 0, 0] ,
"passinsp" : [ 19, 0, 0, 0, 0, 0, 0, 0, 0, 0] ,
"failinsp" : [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] ,
"trigover" : [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] ,
"procover" : [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] ,
"fifoover" : [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] ,
"timeout" : [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
}
totalinsp: Total inspections
passinsp: Passed inspections
```

```
failinsp: Failed inspections

trigover: Trigger overruns

procover: Process overruns

fifoover: FIFO overruns

timeout: Inspection Timeouts
```

URL

```
PUT /api/v1/stats
```

record

This API retrieves an inspection record result. With no report index, the latest report is returned. A specific inspection record can be retrieved by appending the index to the URI. Arguments can be added in the form of argument=value to filter the return set of records that match the filter requirements.

Inspection records will contain the following data in addition to any global data values in use by di-soric Link within that inspection.

cycletime	Cycle time in ms
drawtime	Time used to render graphics for display in ms
failinsp	Cumulative number of failed inspections
idletime	Number of ms the camera was idle between inspections
image	The ID of the image used in this inspection. The SVG or image can be retrieved using the SVG or IMAGE APIs by appending this ID to the request. eg. /api/v1/svg/33
pass	1 if the inspection passed, 0 if failed
passinsp	Cumulative number of inspections passed
recordid	The reference ID for this record. This record can be retrieved individually by appending this ID to the /record request. eg. /api/v1/record/21
timestamp	A timestamp for the inspection in ms
totalinsp	Cumulative total number of inspections

Resource URL to Get Last Record

```
GET /api/v1/record

Accept: application/json
```

Response Body

```
The last completed record in a json array called "records"
```

Example

```
GET /api/v1/record

Accept: application/json
```

Response Body

```
{
        "records":
        [
                {
                        "cycletime": 475,
                        "drawtime": 1,
                        "failinsp": 2,
                        "idletime": 1084,
                        "image": 0x00011eed000a1e1f,
                        "int1": 1,
                        "int101": 40,
                        "pass": 1,
                        "passinsp": 3,
                        "recordid": 21,
                        "timestamp": 8895730.13184849,
                        "totalinsp": 5
                }
        ]
}
```

Resource URL to Get a Specific Record

```
GET /api/v1/record/<recordid>

Accept: application/json
```

Response Body

A single record with the matching record I.D. if it exists; an empty array if it does not.

filters

Records can be filtered by the contents of the record by appending the filter requirements as arguments to the request. A specific value can be specified (eg. int1=4). The symbols (a,b) [a,b] can be used to specify a range of values to include () for a non-inclusive range, [] for inclusive. (and [can be used independently on either end of the range (eg. (4,10] or [5,11) for values 5,6,7,8,9,10)

For example, to include all inspections from record I.D. 5 to 10:

```
GET /api/v1/record?recordid=[5,10]
```

To include recordids from 5 to 10, not including 5:

```
GET /api/v1/record?recordid=(5,10]
```

To include all record I.D.s following record I.D. 5 to an unbounded limit, omit the second argument.

```
GET /api/v1/record?recordid=(5,]
```

Filters can be combined to further limit the returned results. To get all the records between 100 and 200 inclusive that failed only:

```
GET /api/v1/record?recordid=[100,200]&pass=0
```

Special filters to limit the number of returned results start with '!' include !count and !last. When used with a filter range request !count=<x> can limit the number of results returned to x. To return up to the first 5 records starting at record 100:

```
GET /api/v1/record?recordid=[100,]&!count=5
```

Instead of returning the first 5 records matching the filter conditions !last=1 can be used to return the last set of matching records. To return the last failed record only from all records:

```
GET /api/v1/record?recordid=[1,]&!count=1&!last=1
```

image

This API retrieves an image with a default type of PNG. Buffer can be either from an index or most recent.

Note: Image ID is found via the **record** command.

Resource URL

`camera URL /api/v1/image`

retrieves the most recent acquired image.

`camera URL /api/v1/image/<id>`

retrieves image for the buffer id.

parameter	value	example	description	
format	{png | jpg}	format=jpg	JPG or PNG image type	optional, default png

returns an image object of type PNG by default.

Examples

URL

`10.20.1.191 /api/v1/image/0x00011eed000a1e1f`

Response Image

svg

This API retrieves the SVG graphics for a given image ID.

Note: Image ID is found via the **record** command.

Resource URL

```
camera URL /api/v1/svg/<id>
```

Examples

URL

```
10.20.1.191 /api/v1/svg/0x00011eed000a1e1f
```

Response Image

file

This API retrieves a directory listing for the web server directory or sub directory of the web server using a GET request. It also allows for uploading of files via a POST request. If the file uploaded ends with the extension .zip the file will be unzipped in the specified folder.

Resource URL

```
GET /api/v1/file

Accept: application/json
```

retrieves the root file directory of the web server.

```
GET /api/v1/file/<path>

Accept: application/json
```

retrieves image for the sub directory <path>.

Returned Data Values

```
name   Directory or file name

dir    True if a directory, false if a file

size   File size if object is a file, in bytes

POST /api/v1/file

Content-Type:  multipart/form-data
```

uploads a file to the root folder of the web server.

```
POST  /api/v1/file/<path>

Content-Type: multipart/form-data
```

uploads the attached file to the sub directory <path>.

GET Example

URL

```
GET /api/v1/file/app
```

Reply

```
{

    "data": [

        {

            "name": "hmi",

            "dir": true,

            "size": 0

        },

        {

            "name": "img_demo",

            "dir": true,

            "size": 0

        },

        {

            "name": "svg_demo",

            "dir": true,

            "size": 0

        }

    ]

}
```

POST Example HTML Form File Selector

```
<form method="POST" action="/api/v1/file/app/myapp"
enctype="multipart/form-data">

      <input type="file" name="file" />

      <input type="submit" value="Upload" />

</form>
```

POST Example jQuery Ajax Call Using FormData from an HTML Form File Selector

```
<form id="target" enctype="multipart/form-data">

      <input id="file" type="file" name="file" />

      <input type="submit" value="Upload" />  <br/>

</form>

var fdata = new FormData($('form')[0]);

$.ajax({

      url: '/api/v1/file/app/myapp',

      type: 'POST',

      data: fdata,

      cache: false,

      contentType: false,

      processData: false,

      success: function(data, textStatus, jqXHR){

      }

});
```

Chapter 5 REST APIs

6

CHAPTER 6

Installing and Updating CloudLink

This section describes how to install and update CloudLink on a CS50 Sensor for use with CS50 Software.

Installing CloudLink

The CloudLink web interface can be installed on the CS50 Sensor using the **Firmware Update Tool**.

A .zip file containing CloudLink files is placed on the PC during CS50 Software installation, and that .zip file is pushed to the sensor during a firmware upgrade using the Firmware Update Tool.

The Firmware Update Tool sends the. zip file by FTP along with the other firmware files. The sensor extracts the files from the .zip file as part of the upgrade process.

Note: The CloudLink .zip file is sent by FTP when the sensor is being upgraded.

CS50 Sensor firmware and CloudLink install in six stages:
• Transfer;
• Prepare;
• Apply DSP Update;
• Apply Kernel Update;
• Apply Web Interface Update (CloudLink Update);
• Reboot.

The images on the following pages show examples of a CS50 Sensor's simultaneous CloudLink and firmware installation process.

Apply DSP Update

Smart Camera Firmware Update Tool

Please Select a Smart Camera to Update

125ADF

Model	WVGA
Firmware Version	**Build 1 UPDATE RECOMMENDED**
MAC Address	00:0B:43:12:5A:DF
IP Address	10.20.1.113 [mask: 255.255.255.0]
Status	Updating Firmware
PC Version	**Build 0**

Select the Firmware Version

Build 0

Applying DSP Update (Stage 3 of 6)

Do not power down or disconnect camera!!!

Exit   Update Firmware

Apply Kernel Update

Smart Camera Firmware Update Tool

Please Select a Smart Camera to Update

125ADF

Model	WVGA
Firmware Version	**Build 1 UPDATE RECOMMENDED**
MAC Address	00:0B:43:12:5A:DF
IP Address	10.20.1.113 [mask: 255.255.255.0]
Status	Updating Firmware
PC Version	**Build 0**

Select the Firmware Version

Build 0

Applying Kernel Update (Stage 4 of 6)

Do not power down or disconnect camera!!!

Exit   Update Firmware

Apply Web Interface Update (CloudLink Update)

Smart Camera Firmware Update Tool

Please Select a Smart Camera to Update

125ADF

Model	WVGA
Firmware Version	**Build 1 UPDATE RECOMMENDED**
MAC Address	00:0B:43:12:5A:DF
IP Address	10.20.1.113 [mask: 255.255.255.0]
Status	Updating Firmware
PC Version	**Build 0**

Select the Firmware Version

Build 0

Applying Web Interface Update (Stage 5 of 6)

Do not power down or disconnect camera!!!

Exit   **Update Firmware**

Updating CloudLink

CloudLink is automatically updated. It is also possible to update CloudLink by itself without updating other sensor firmware. This is done by pressing the **Control-W** key combination while selecting the sensor to update from the dropdown list.
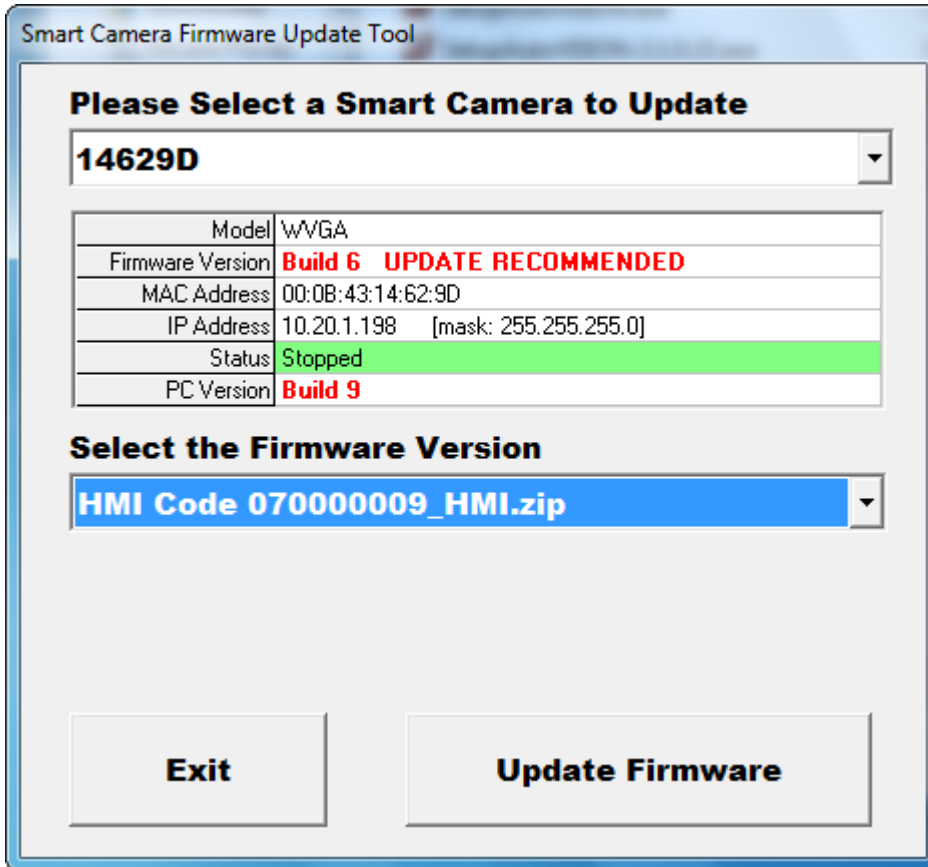
Once the sensor has been selected, choose the desired version of CloudLink in the **Select the Firmware Version** dropdown menu and then click the **Update Firmware** button. A sensor reboot is not necessary after the update and the unpacked CloudLink files are immediately ready for use.

A

APPENDIX A
Changing the CloudLink HTTP Port for PC-Based Systems

This section contains information about how to change the CloudLink HTTP port for PC-based systems.

Changing the CloudLink HTTP Port

The CloudLink HTTP port for PC-based systems can be changed to a value other than 8080 by adding a registry key. This is useful when software is installed on PCs that are already using port 8080 for an application other than CloudLink.

Important: This function is for PC-based systems only. Sensors are fixed at port 80.
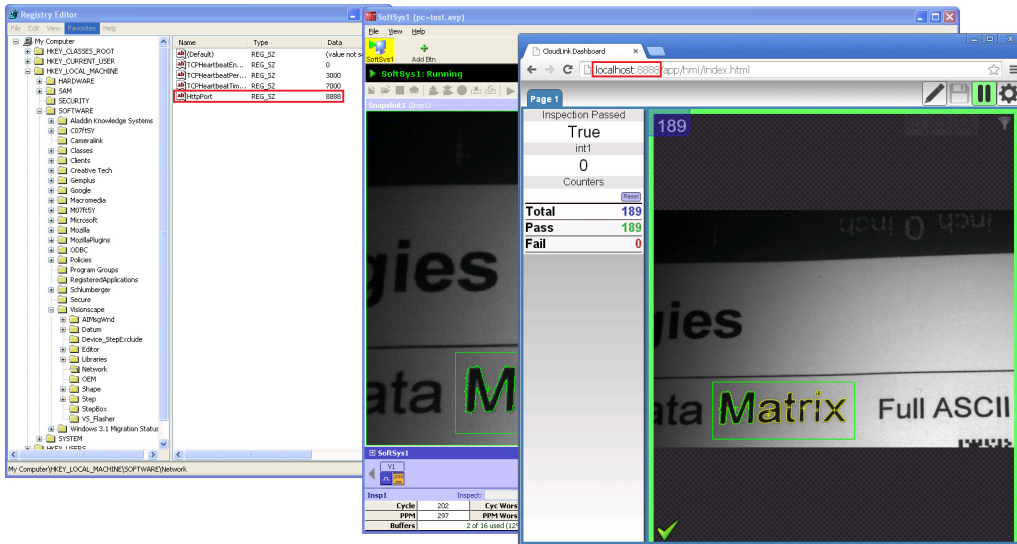
For 32-Bit Systems

To change the CloudLink HTTP port to a value other than 8080, add the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Visionscape\Network\[HTTP Port]

The **[HTTP Port]** value must be a string containing the desired port number.

You must also cycle the Visionscape Backplane when changing the server setting.

The example below shows the HTTP Port changed from 8080 to **8888**. Notice that the **localhost** address now shows 8888 as the port number.

For 64-Bit Systems

To change the CloudLink HTTP port to a value other than 8080, add the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Visionscape\Network\[HTTP Port]

The **[HTTP Port]** value must be a string containing the desired port number.

You must also cycle the Visionscape Backplane when changing the server setting.

As in the example above, the **localhost** address will show the new port number.