Star CloudPRNT

Developer Guide

STAR MICRONICS CO.,LTD.

Rev 1.4

Star cloudPRNT Developer Guide

1. Introduction

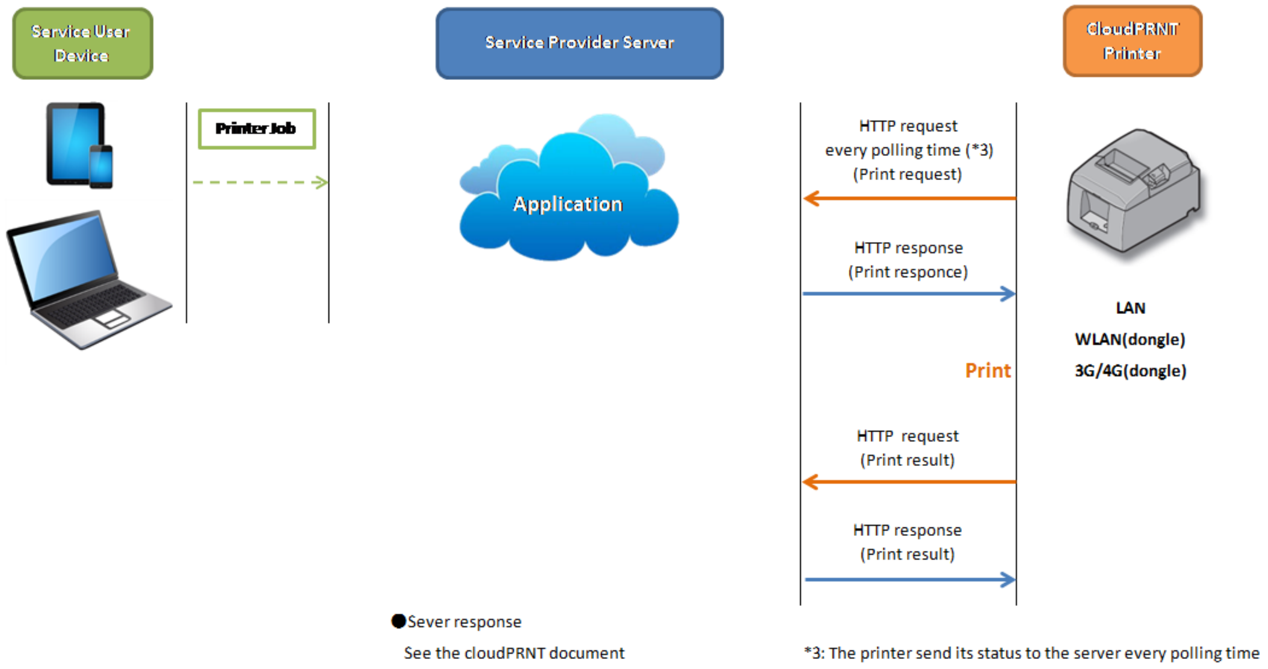
Star CloudPRNT is a protocol to enable printing from remote servers. Customers are required to implement a server following this protocol to enable printing to remote devices.
CloudPRNT is designed to be simple to implement, versatile and secure. By using common http/https to pass a REST/JSON API and common print job data formats, it is a familiar technology for web developers and does not requires specific firewall, port forwarding or tunneling to enable connectivity.

Overview

All CloudPRNT communication is handled via a single URL, which can be hosted by any web server. Clients (such as the HI01/02X Intelligent Interface), use an http POST method, to update device status and make requests at a fixed interval, the server will respond with instructions and availability of data for printing. Clients will then use an http GET method to retrieve print jobs, in the requested data format. Finally, clients will respond with a DELETE or POST with error code to inform the server of the printing status.

All negotiations use JSON encoded information, except for the print job request, which will use an http GET (with URL encoded parameters), and will deliver the job in the requested data format.

Printing Process


Service User Device
Printer Job
Service Provider Server
Application
CloudPRNT Printer
HTTP request every polling time (*3) (Print request)
HTTP response (Print responce)
Print
HTTP request (Print result)
HTTP response (Print result)
LAN
WLAN(dongle)
3G/4G(dongle)
●Sever response
See the cloudPRNT document
*3: The printer send its status to the server every polling time

Operating Environment

The Star CloudPRNT client service is implemented on the Star Intelligent Interface (IFBD-HI01X and IFBD-HI02X), with support for the following printers.

*Firmware: Ver. 1.1.0 or higher

Interface Board model	Printer model	Minimum Printer Firmware		Minimum Interface Firmware	Note
		Boot Version	Main Version		
HI01X	TSP650II	1.0	1.0	1.1	-
HI01X	TSP700II	2.0	3.0	1.2	-
HI01X	TSP800II	1.0	1.2	1.2	-
HI02X	SP700	2.0	3.0	1.2	-

- HTTP and HTTPS (TLS 1.2) are supported.
  Note: Common “HTTP Basic Authentication” allows username/password security.
- All CloudPRNT communication is handled via a single URL.
- For USB peripheral support, HI01x/HI02x firmware version 1.4 or later is required.

2. Client configuration

Each client must be configured with the CloudPRNT URL that it should poll. It is also be possible to provide a username/password for authentication and polling interval. The username and password are optional settings, depending on the requirements of the server.

If a server developer requires further information to be provided by the client, then these should be passed by adding a query string to the polling URL.

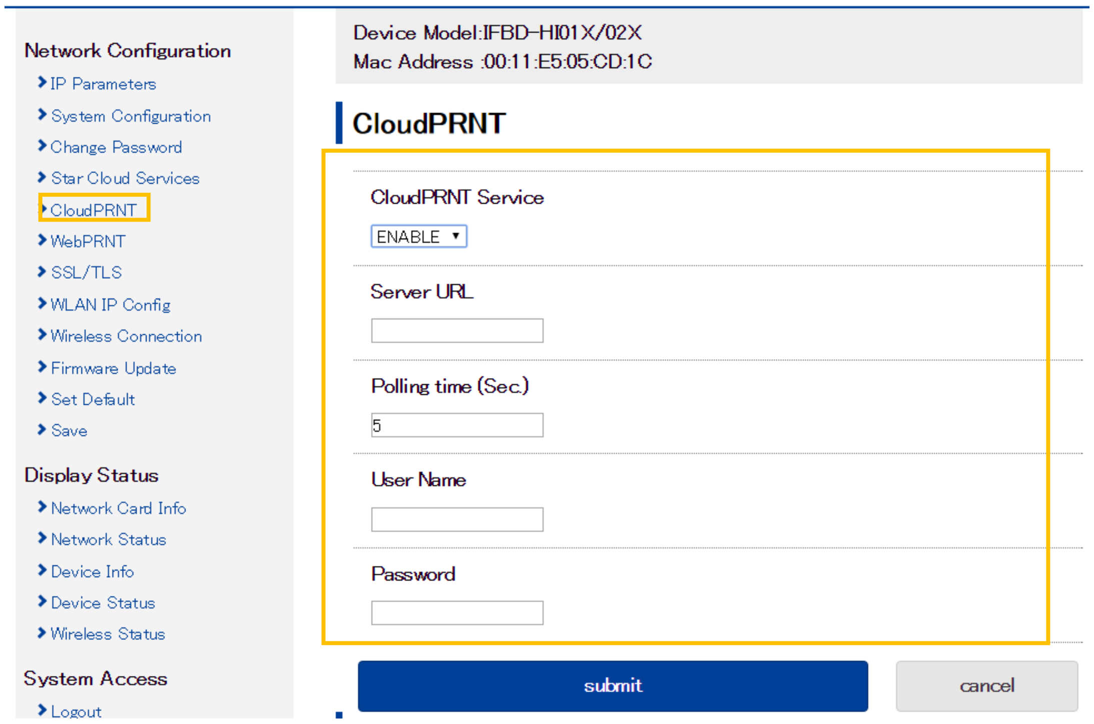
These options are set through the web based configuration user interface of the HI01X/HI02X interface board.

Configurable client options are:

- Poll URL (required) - a URL that the client will poll regularly through an http POST
- Poll Interval (optional) - poll timer in seconds. The client will connect to the server at this interval to provide the server with live status updates, and check for print jobs or client action requests. It not specified, then the client will use the default 5 seconds.
- User (optional) - a user name that will be passed in the first request body if specified.
- Password (optional) - a password that will be specified in the first request body, if specified.

The HI01X/02X supports https/TLS web services, by default servers are verified using the same CA Root set as Mozilla Firefox. Users may also install custom CA Certificate set to the HI01X/HI02X interface.

Reference: Web configuration display of the HI01/02X.

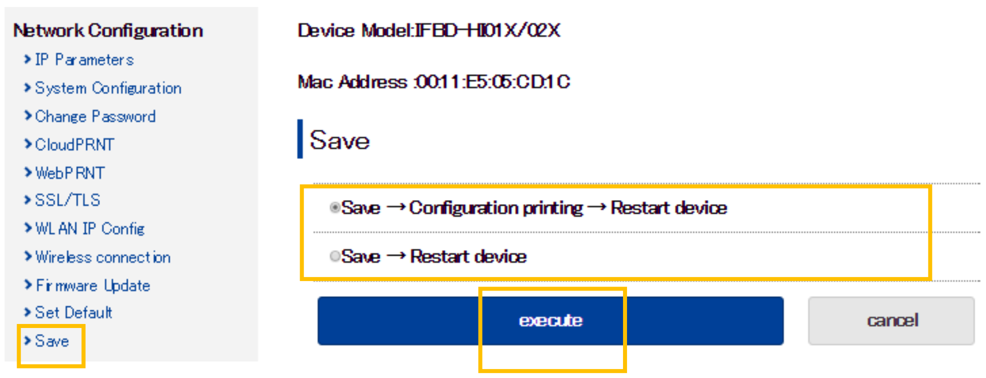


Device Model:IFBD-HI01X/02X
Mac Address :00:11:E5:05:CD:1C

CloudPRNT

CloudPRNT Service: ENABLE
Server URL:
Polling time (Sec.): 5
User Name:
Password:

Note: After CloudPRNT setting, you must do "save" to apply your setting to HI01/02X.
Then the printer will reboot to complete it.



Device Model:IFBD-HI01X/02X

Mac Address :00:11:E5:05:CD:1C

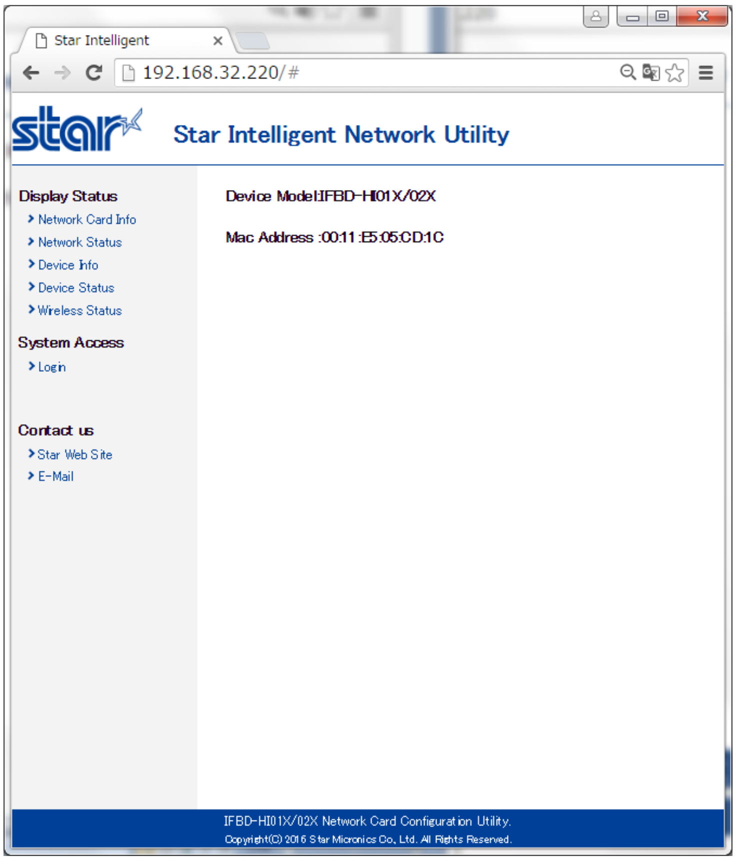
Save

[✓ Save → Configuration printing → Restart device]
Save → Restart device

Note: All HI01/02X configuration is performed via a user interface hosted on the built-in web server.

1. Check the IP address of the HI01/02X -Printer by self-print.
   - Connect Ethernet cable to the HI01/02X -Printer.
   - Turn on the HI01/02X -Printer with pushing the feed button for 5 seconds.
   - The IP address of the HI01/02X -Printer is shown in 2nd print paper.

2. Put IP address of HI01X into the web-browser. Then the web configuration utility is shown.



Device Model:IFBD-HI01X/02X

Mac Address :00:11:E5:05:CD:1C

IFBD-HI01X/02X Network Card Configuration Utility.
Copyright(C) 2016 Star Micronics Co., Ltd. All Rights Reserved.

3. To enter the setting of HI01 page, click “Login” and put user name and password.
   User name: “root” , Pass word: ”public”.

3. Polling the server (POST)

On each poll interval the client will send an http POST to the server. The POST is used to send live information to the server. The server response is then used to inform the client of any available print jobs, or to provide requests for device management features.
Devices with support for additional peripherals, such as USB barcode readers, displays, keyboards or scales, with also report the availability and status of these devices.

Poll Timing

When idle, the device will poll the server based on the configured polling interval. With the following exceptions:

- If an event occurs, such as a printer status change, a barcode being scanned or a key being pressed, then a poll to the server will be triggered immediately. This allows very low latency operation, with the server able to respond to user events very quickly.
- While a print job is in progress, the device may not send polls to the server until the job is complete. This is dependent on device implementation, and typically devices that support operations other than printing (such as barcode scanning) will continue to poll the server as usual, even while printing.

  Compatibility Note

  Versions up to 1.3 of HI01x/HI02x firmware do not poll while printing. However version 1.4 or later does send poll requests to the server during print operations, this allows continuous usage of all peripherals during a print operation.

JSON Request

The request body sent from the client is in JSON format, based on the following template:

```
{
     "status": "<ASB Hex format>",
     "printerMAC": "<Ethernet MAC address>",
     "uniqueID": "<server assigned ID>",
     "statusCode": " <description>",
     "printingInProgress": bool,
     "clientAction": [{
         "request": "<request type>",
         "result": "<request result>"
     }],
     "barcodeReader": [{
         "name": "<device logical name>",
         "status": { "connected": bool, "claimed": bool },
         "scan": [ {"data": "<scanned barcode character sequence>",
                    "symbology": "<symbology of scanned barcode>"}]
     }],
     "keyboard": [{
         "name": "<device logical name>",
         "status": { "connected": bool, "claimed": bool },
         "keyPresses": "<key presses since last poll>"
     }],
     "scale": [{
         "name": "<device logical name>",
         "status": { "connected": bool, "claimed": bool },
         "weight": [{ "data": "<last read weight>",
                      "unit": "<Unit of last weight kg/g/lb>",
                      "stable": bool }]
     }],
     "display": [{
         "name": "<device logical name>",
         "status": {"connected": bool}
     }]
}
```

All fields, except "statusCode" are optional and will be specified only if configured and needed, or may have a null value, servers should be able to handle either case..
The "clientAction" field is specified only if the server has sent a "clientAction" request to ask for specific details, (e.g. firmware version, memory switch setting etc) or to inform the server of events that are not encoded in printer device status.

- "status" - provides the server with printer status information in Star ASB style. If this is not provided, then the server can assume that the printer is online if statusCode value beginning with "2" (e.g. "200" is provided).
- "printerMAC" - provide the server with the MAC address of the printer. Note that this is primarily for

client identification purposes and may not be the MAC address that the request is issued from in case the client has multiple routes to the server. For example, the HI01/02x interface with always use the Ethernet MAC address, even if WiFi is used.

- "uniqueID" - a server assigned ID. This field will only be included in the request if the server has assigned an ID.
- "statusCode" - provide a 3 digit status code (based on http status codes, but not directly compatible) and description of the printer state. Typically this will be "200 OK" to indicate that the printer is online and ready. In case of print job failure, or offline, then an error state can also be passed, see Printer Status Codes section for details. Status codes from a cloudPRNT client are URL encoded, because they may also be sent as a URL parameter, for example when the printer sends a DELETE request.
- "printingInProgress" – **added in HI01x/HI02x firmware version 1.4.** A Boolean value which indicates that a print operation is currently in progress. This is provided by clients which may issue polls to the server during a print operation. This is helpful to indicate that the “jobReady” response field will be ignored and so the server may avoid doing database or similar checks to determine whether a job is available. It may also be useful status to display to users.
  Please note that this field is optional and may not be included in the POST request from all clients, therefore it should not be used as a signal that printing is completed (please monitor GET/DELETE and POST responses for this).
- "clientAction" – an array of responses to server requests, see Client Action section below. If there are no responses needed, then this field may be missing, of have a value of null.
- "barcodeReader" - **added in HI01x/HI02x firmware version 1.4.** An array of objects representing available barcode reader hardware. Please see **Barcode Reader** section below.
- "keyboard" - **added in HI01x/HI02x firmware version 1.4.** An array of objects representing available keyboards. Please see **Keyboard** section below.
- "scale" - **added in HI01x/HI02x firmware version 1.4.** An array of objects representing available scales. Please see **Scale** section below.
- "display" - **added in HI01x/HI02x firmware version 1.4.** An array of objects representing available line displays. Please see **Display** section below.

Example POST Data

Common poll data, includes printer status and MAC address identifier:

```
{
    "status": "23 6 0 0 0 0 0 0 0 ",
    "printerMAC": "00:11:e5:06:04:ff",
    "statusCode": "200%20OK",
    "clientAction": null
}
```

Poll sent after the server has requested the polling interval and list of supported encodings using "clientAction" requests, to a printer with an "UniqueID" set to "Star1":

```
    {
        "status": "23 6 0 0 0 0 0 0 0 ",
        "printerMAC": "00:11:e5:06:04:ff",
        "uniqueID": "Star1",
        "statusCode": "200%20OK",
        "clientAction": [
            {"request":"GetPollInterval","result":"10"},
            {"request":"Encodings","result":"image/png; image/jpeg; application/vnd.star.raster;
application/vnd.star.line; application/vnd.star.linematrix; text/plain; application/octet-stream"}
        ]
    }
```

Response

The server will respond with a JSON encoded reply, based on the following template:

```
{
     "jobReady": true¦false,
     "mediaTypes": [ "<content media type>" ],
     "deleteMethod": "DELETE|GET",
     "clientAction": [ {"request": "<request type>", "options": "<request parameters>"} ]
     "claimBarcodeReader": [ "<device name" ],
     "claimScale": [ "<device name>" ],
     "claimKeyboard": [ "<device name>" ],
     "display": [ { "name": "<device name>", "message": "<message markup>" } ],
     "reportUnstableWeights": [ "<device name" ]
}
```

The server response informs the client of any pending print data, and also allows the server to pass Client Action requests to the client.

- "jobReady" - if "true" then the server has data for the client which should be printed, it can be retrieved using an http GET request.
- "mediaTypes" - a list media (MIME) types that available print data may be provided in by the server. For example a server may be able to serve a document as a PNG image, or convert to Star Raster. The client will choose it's preferred format from those available. Note that this can vary between print jobs, for example a server may handle printing bit image and text data types differently.
- “deleteMethod” – ask the printer to confirm print job completion with either an http DELETE or GET request. This field is optional, and the printer will default to using a DELETE.
  Generally, supporting DELETE is recommended, as the standard way to remove a resource from a web server. However, some web servers are unable to pass DELETE request to CGI scripts, and so the printer will optionally send an http GET, with specific query string parameter instead.
- "clientAction" – an array of requests for a special action from the client, see Client Action section below.
- "claimBarcodeReader" – An array of strings, each string is the logical device name of a Barcode Reader device to be claimed for input. Alternatively, a Boolean response of ‘true’ can be sent to claim the ‘default’ barcode reader.
  Claiming a barcode reader means that scanned barcodes will be reported to the server. The claim lasts only until the next poll, therefore the server should set this field every time if it wishes to receive barcode scans continuously.
- "claimScale" – An array of strings, each string is the logical device name of a Scale device to be claimed for input. Alternatively, a Boolean response of ‘true’ can be sent to claim the ‘default’ scale. Claiming a scale means that and change in weight measurement will be reported to the server. The claim lasts only until the next poll, therefore the server should set this field every time if it wishes to

receive weight changes continuously continuously.

- "claimKeyboard" – An array of strings, each string is the logical device name of a Keyboard device to be claimed for input. Alternatively, a Boolean response of 'true' can be sent to claim the 'default' keyboard.
  Claiming a keyboard means that key presses will be reported to the server. The claim lasts only until the next poll, therefore the server should set this field every time if it wishes to receive keyboard activity continuously.
- "display" – A string containing a message for the default display, or an array of display messages. Each message should contain two string fields:
  - "name" – the name of the display device to send the message to.
  - "message" – The message to be sent to the display, this can be plain text with CloudPRNT display markup messages, Refer to the **Display** section below for details.
- "reportUnstableWeights" – Enables reporting of unstable weights from scale hardware. This field can be an array of strings, with each string containing the logical name of the scale for which unstable weight reporting is required. Alternatively, this may be a boolean value, if set to true, then unstable weight reporting will be enabled for the default scale.

To simplify the client←→server rules, a server should not set a clientAction request and expect printing at the same time. If "jobReady" is set to true, and one or more clientActions are requested, the client will handle the clientAction requests only. It is expected that the printing can wait for the next poll. This does not usually cause any visible performance penalty, because the client will issue the next poll immediately after handling the clientAction requests instead of after the usual polling interval.

Example server POST responses

A typical server response, when no printing is required:

```
{
     "jobReady": false
}
```

A typical server response when printing is required, and the server is able to make the job available as either plain text, or png data.

```
{
     "jobReady": true,
     "mediaTypes": ["text/plain", “image/png”]
}
```

A server response, with request for the printer to report its polling interval and list of supported job encodings:

```
{
     “jobReady": false,
     "clientAction": [
              {"request": "GetPollInterval", "options": ""},
              {"request": "Encodings", "options": ""}
     ]
}
```

Client action

A "clientAction" option in the response packet is used by the server to ask the client to perform an action other than printing, or to return information that is not typically included inside the normal client POST Request packet. If the server issues a "clientaction" request, then the client will send a POST Request in reply as soon as it has handled this action, instead of at the usual poll interval.

For performance, multiple client actions can be passed as an array, allowing several requests to be made simultaneously. Clients can also respond with an array, and in some cases may generate several responses. For example a firmware update may issue an action at the start, and the end of the update process.

Normal printing operation is entirely possible without requiring any client action features.

Version	ClientAction Request	Parameter	Client Request result
1.0	ClientType		Responds with a fixed string indicating the client type. Pre-defined types are: "Star Intelligent Interface HI01X"
1.0	ClientVersion		Responds with the cloudPRNT version implemented by the client. currently this will always be "1.0.x". Where 1.0 represents the cloudPRNT specification version, and "x" the implementation revision.
1.0	Encodings		Returns a semi-colon separated list of supported print job content encodings supported by this client. The encoding name follows the common http mime/content type string format.
1.0	SetID	New ID string.	Set the ID that the client will use within the "UniqueID" field of POST/GET requests
1.0	GetPollInterval		Report the clients polling interval in seconds. This may be used by the server to determine a safe timeout for determining a loss of client connection.

1.4	PageInfo		Request printer page information, which will be returned as a JSON formatted object, containing the string fields:
			{
			"paperWidth": "<width in mm>",
			"printWidth": "<print width mm>",
			"horizontalResolution": "<dots per mm>",
			"verticalResolution": "<dots per mm>"
			}
			These are returned as a string to avoid rounding errors.

Note, Client Actions are technically optional, and may not all be supported by all clients. However, all above client actions are fully supported by the Star HI01X and HI02X printer interface cards.

4. Print job requests (GET)

Request (GET)

When an http GET method is used, it is intended to pull a print job from the server, which will then be printed. Please note that, in accordance with the http specification, GET should have no server side effects, simply re-sending the same GET should result in re-downloading the same job until the server state is changed by a POST or DELETE. A server can assume that the print is in progress after the GET, until a DELETE, or POST with error code is received.

When the Client has been informed (via a POST response) that print data is available, then it will retrieve this through an http GET request. In accordance to the http specification, the GET does not have a message body, but instead passes all parameters via a query string, or inside header options.

```
[http/https]://[cloudprntURL]?uid=<printer ID>&type=<media type>&mac=<mac address>
```

- **uid** is a server specified UniqueID. This will only be included if it has been set by the server.
- **type** is the requested media data type of the print job. Servers may serve jobs in multiple forms if they prefer
  (e.g. Star Line, Raster, ESC/POS, PNG, plain text). If so then the printer will choose the type that it prefers to handle from those available. If this is parameter omitted, then the server should return data in its preferred media format.
- **mac** is the printer MAC address, as used in the POST request.

Servers can consider the print job to be in progress after the data is retrieved by a GET. Clients may optionally stop POST based polling while printing (after the GET), until the print is complete, or an error has occurred. It is also possible that subsequent GET requests may be sent, for example if the printer chooses to retry with a different mime type after a data decoding issue.

Response

The server response should include the print job data as the message body, encoded in the requested format, and with the requested format as the specified content type in the http header. It is possible that the client will re-request the data, by sending further GET requests for the same document - this is unlikely in usual operation but should be supported by the server.

The server should set an appropriate "Status: " field in the response header, with one of the following values.

· **200 (OK)** If the operation succeeded and the job was returned as expected.
· **404 (not found)** If no data is available for printing.
· **415 (unsupported media type)** if the request was for a content type that the server can not support.

In case of an error status code, the response body will be ignored, and so may be empty.

When a print job uses the Media type which is **text/plain, image/png, image/jpeg**, then the server can set extra control options in the response header(*), with one of the following field and value.
(*)This function has been supported since F/W version **1.3 or later**.

Header Field Name	Example	Description
X-Star-Buzzerstartpattern	X-Star-Buzzerstartpattern: 1	Performs buzzer before printing the sent job. The specified value is a number of performing buzzer. Value Range: From ‘1’ to ‘3’
X-Star-Buzzerendpattern	X-Star-Buzzerendpattern: 1	Performs buzzer after printing the sent job. The specified value is a number of performing buzzer. Value Range: From ‘1’ to ‘3’
X-Star-Cut	X-Star-Cut: full; feed=true	Specified the method of cut to perform at the end of a document. Support for this header has been added since firmware version 1.4. Supported Values: full, partial, none. Feed option may be set to true or false, but defaults to

		true of not specified.
X-Star-ImageDitherPattern	X-Star-ImageDitherPattern: none	Specify the dithering method to use when printing and image (defaults to 'fs' if not specified) Values: • none – disable dithering • fs – use Floyd Steinberg error diffusion dither
X-Star-CashDrawer	X-Star-CashDrawer: start	trigger opening a cash drawer at the start or end of a job. Values: none, start, end

5. Print job confirmation (DELETE)

Clients will confirm job completion with the server. This is handled by issuing an http DELETE request to inform the server to clear the current job. The query string is equivalent to the http GET request, but with the addition of a "code" parameter, and no need for the "type" parameter. A DELETE may be used by the client to clear a completed print job, or to clear a job that it is unable to process, the server should check the 'code' parameter to determine the difference.

Print Success

In case printing completes correctly, the client will send a DELETE to the server, with the "code" parameter set to "OK", the "pid" and "mac" parameters will be the same as the preceding GET. Next, the client will begin usual polling through POST requests.

Print failure or error

In case printing fails, or an error occurs during printing, the client will follow the sequence:

1. Send a POST request to the server with a relevant printer status code. If status is not OK (beginning with a "2"), then the server can determine that printing did not succeed.
2. If the client determines that the failure is not related to the print data, then it will enter the normal polling process. After the printer issue has been resolved and it is online again, it will attempt to GET the job again if the server is still reporting that job data is available.
3. If the client determines that printing failed because it is not able to handle the active print job data, then it will send a DELETE, with corresponding code value, as described in Printer Status Codes. After the DELETE, the client will resume the usual POST based polling process.

Alternative Print job confirmation (GET)

Some web servers do not support DELETE requests through server side CGI scripts. Therefore, an alternative method is provided. If the server requests this mode (see poll response section above) then the printer will send an http GET request instead of a DELETE.

This request can be differentiated from a standard job download GET request, because the printer will add “delete” as a query string parameter. In all other respects, the query string and behavior are the same as for the usual DELETE request.

6. Content Media Types

In general, cloudPRNT follows the common internet MIME content type names for print job data such as JPEG, PNG, text. However, cloudPRNT does support some original data formats and for this we will introduce new content type string names.

Data Format	Media Type
Star Thermal Line Mode	application/vnd.star.line
Star Matrix Line Mode	application/vnd.star.linematrix
Star Raster	application/vnd.star.raster

The HI01/02X interface supports the following media types:

text/plain, image/png, image/jpeg, application/vnd.star.line, application/vnd.star.linematrix, application/vnd.star.raster, application/octet-stream

Note: if sending raw command data for printing then it is not recommended to use application/octet-stream. Instead please use the appropriate vnd.star.* content types. This way the interface can select the most appropriate available type and can avoid attempting to write incompatible command data to the printer.

Image Printing

The HI01/02X interface can support PNG encoded print job data. When the server sends an image based print job, by default the image will be printed pixel for pixel, without scaling. Therefore the server should send image data matched to the printers resolution (576 pixels wide on 80mm printer models, 832 pixels on 112mm printer models). By default, the image will also be dithered when converted to monochrome for printing.

Text Printing

The HI01/02X interface can support plain text print job data. In this case the text data will be output with the printers default font and codepage, followed by a cut. It is not recommended to embed control code sequences, as this is not strictly plain text data. If control code sequences are required then please specify the emulation by using an “application/vnd.star.*xxxx*” content type.

Compatibility Note: Version 1.1 firmware for HI01X interfaces, could support a media type of “text/plain”, but did not accept variations with additional content information, such as “text/plain; encoding=utf-8” which is automatically appended by several server-side development solutions.
From version 1.2 firmware (for HI01X or HI02X interfaces), content media types with a semi-colon to separate type from additional content information are supported.

7. Printer Status codes

Printer status codes are similar to http status codes, but not directly compatible. They are a 3 digit error code, which can be followed by a space, and then an error description. Note that error descriptions are intended for logging, and may vary between clients. Error codes will be added to this document as required, but follow a pattern meaning that a server need not recognise all specific error codes. For example, all codes beginning "2" (200, 211, etc.) mean that the printer is online, all codes beginning with a "4" represent a printer error, etc.

Code	Description
2xx	Printer is online and able to print
21x	Printer is online, but a paper related warning is n place, the printer may go offline in the near future.
210	Paper low
4xx	Printer error - all codes beginning with 4 indicate a failure due to printer error.
41x	Paper error.
410	Out of paper.
411	Paper jam.
42x	Cover open error.
420	Cover open
5xx	Client error - all codes beginning with 5 indicate a client issue.
51x	Media compatibility error
510	Incompatible media type, client does not support the data issued by the server.
511	Media decoding error. Client supports the media type, but failed to decode it. May indicate a data corruption, or differing versions of the media format.
512	Unsupported media version. Means that the media type is supported, but the version issued by the server is not compatible. E.g. client may support PDF up to 1.4, but not 1.5.
52x	Job download error
520	Timeout, client could not download the job within an internal timeout limit.
521	Job too large. The job data is too large for the client's download buffer or exceeds a specified size limit.

8. General Server Error Responses

At any time, the server ,may respond with the following status responses in case of error conditions.

- **400 (bad request)** if the request is not understood.
- **401 (unauthorised)** if the user/pass are incorrect. Or if the server wishes to refuse for another reason.
- **403 (forbidden)** if the server rejects the request for a reason other than authentication, such as a network security issue (e.g. IP or MAC is not in an allow list).
- **429 (too many requests)** If the client is making requests too frequently, then the server may start rejecting them. This may mean that the client is not obeying the requested polling time.

9. Connection Timeout

Clients choose their interval, but this may vary between differently configured clients. Clients are permitted to send requests more frequently in the case that their status has changed several times quickly, or in response to a server "clientAction" request. However, it is never expected that the client will poll significantly more slowly than requested.

Therefore, a long delay between polls can be considered as a warning that the printer has been powered off, or the network connection is lost. The server may then report this and take appropriate action (anything from displaying "offline/disconnected" to emailing a technician, this is server dependent).

The precise timeout is not always easy to calculate, especially while printing a job, which may take several seconds, or event minutes.
Therefore, it is recommended that a server calculate two timeouts for a particular job queue.

- **Poll timeout** recommended to be 2 * <poll interval> + 5s - this specified the time between polls before the server assumes that a device has gone offline or lost its connection.
- **Print timeout** recommended to be around 60s - this is the time allowed for a printer to accept and print a job before completing with a jobstate "restart"/"delete" before the server can assume that the printer has gone offline. The real time to print a document can vary significantly depending on the printer performance, job length and type of data, therefore it is recommended that a server considers its own usage scenario and adjusts this timeout accordingly.

Some devices will continue to send status poll requests during job printing, to verify that they are still online. In this case the Print Timeout is not essential, but not all clients can be guaranteed to do this. With firmware versions up to 1.3 the HI01X and HI02X interfaces will not issue POST requests after requesting a print job, until the job is complete or has failed, however version 1.4 and later will issue POST requests while printing is in progress.

10. Security

HI01X/HI02X, clients optionally support the HTTP Basic Authentication standard, for username/password authentication and https/TLS encryption. In situations where authentication is needed, customers are strongly recommended to use TLS encryption also, to avoid easy password capture through monitoring unencrypted http traffic.

The Star HI01X/HI02X hardware includes a set of common CA certificates which are synchronized with the current Mozilla recommendations. As a result, it will automatically trust https sites signed with certificates from common Certificate Authorities. However, it is also possible to install a custom CA certificate set if required.

11. Controlling Peripheral Devices

From HI01x/HI02x version 1.4 firmware onwards, it is possible to control externally connected peripheral devices, for full POS functionality. The HI01x hardware uses an internal claiming system to ensure that all devices can be shared reliably between CloudPRNT, and other systems such as WebPRNT.
For Claiming purposes, devices fit into two categories:

- Input devices: these are Barcode Readers, Scales and Keyboards. These are devices that will report events as input, and must be claimed by the application or service that needs to receive this input.
- Output devices: these are the Printer and Line Display. These devices are controlled by sending transactions, and will be automatically claimed and released when used through CloudPRNT.

Barcode Reader

Barcode Readers are input devices, which must be claimed in order to receive scan event information. Barcode information is reported inside the usual POST requests from the client to the server, and can be claimed and released through the server response.
Inside the JSON POST, a “barcodeReader” field will provide an array of JSON objects in the following format:

```
{
    "name": "<device logical name>",
    "status": { "connected": bool, "claimed": bool },
    "scan": [ {"data": "<scanned barcode character sequence>"} ]
}
```

- “name” – specifies a logical device name to uniquely identify the barcode reader. This name may be used to claim the reader in order to receive scan events.
- “status” – an object describing the device status, containing two fields:
    - “connected” – a Boolean value, true if the barcode reader is physically connected and false otherwise.
    - “claimed” – true if the device is claimed by CloudPRNT, meaning the scan events will be reported to this server.
- “scan” – an array of scan events objects since the last POST. Each object contains the following fields:
    - “data” – a string containing base64 encoded data, reported by the barcode scanner.

Keyboard

Keyboards are input devices, which must be claimed in order to receive scan event information. Key press information is reported inside the usual POST requests from the client to the server, and can be claimed and released through the server response.

Inside the JSON POST, a “keyboard” field will provide an array of JSON objects in the following format:

```
{
    "name": "<device logical name>",
    "status": { "connected": bool, "claimed": bool },
    "keyPresses": "<key presses since last poll>"
}
```

- “name” – specifies a logical device name to uniquely identify the keyboard. This name may be used to claim the keyboard in order to receive key press events.
- “status” – an object describing the device status, containing two fields:
    - “connected” – a Boolean value, true if the keyboard is physically connected and false otherwise.
    - “claimed” – true if the device is claimed by CloudPRNT, meaning key press events will be reported to this server.
- “keyPresses” – a base64 encoded string providing the sequence of key presses since the last POST. Key presses are already converded from keyboard scan codes to Ascii data.

Scale

Scales are input devices, which must be claimed in order to receive weight measurement event information. Weight changes are reported inside the usual POST requests from the client to the server, and can be claimed and released through the server response.

Inside the JSON POST, a “scale” field will provide an array of JSON objects in the following format:

```
{
    "name": "<device logical name>",
    "status": { "connected": bool, "claimed": bool },
    "weight": [{ "data": "<last read weight>",
                 "unit": "<Unit of last weight kg/g/lb>",
                 "stable": bool }]
}
```

- “name” – specifies a logical device name to uniquely identify the scale. This name may be used to claim the scale in order to weight change scan events.
- “status” – an object describing the device status, containing two fields:
    - “connected” – a Boolean value, true if the scale is physically connected and false otherwise.
    - “claimed” – true if the device is claimed by CloudPRNT, meaning weight change events will be reported to this server.

- “weight” – an array of scan events objects since the last POST. Each object contains the following fields:
  - “data” – a string containing the reported weight. Reported as a string to avoid potential rounding errors.
  - “unit” – a string indicating the unit of measurement being reported, such as “kg”, “lb”. Possible units depend on the scale hardware.
  - “stable” – a Boolean which will be ‘true’ if the weight measurement is considered to be stable (i.e. no fluctuation) of ‘false’ if the weight is not yet considered to be stable. By default, cloudPRNT will report only stable weight changes, but this can be specified in the POST response in case the server wishes to receive all changes.

Display

Displays are output devices, which do not require claiming. Status information will be reported inside the POST, and display messages can be passed back by the server inside the JSON POST response.

Inside the JSON POST, a “display” field will provide an array of JSON objects in the following format:

```
{
    "name": "<device logical name>",
    "status": {"connected": bool}
}
```

- “name” – specifies a logical device name to uniquely identify the keyboard. This name may be used to claim the keyboard in order to receive key press events.
- “status” – an object describing the device status, containing two fields:
  - “connected” – a Boolean value, true if the keyboard is physically connected and false otherwise.

Sending messages to the display, is handled by specifying a “display” field in the JSON POST response form the server back to the client. Messages are strings which can use a simple markup format.

Command	Function
[cls]	Clear display, and reset cursor to home position
[home]	Reset cursor to home position
[nl]	New Line, moves cursor to the beginning of the next line
[http://..] [https://..]	Display image from URL. The cloudPRNT device will download a PNG or Jpeg image from the specified url and draw to the display (If image display

	is supported).
[enc <encoding>]	Set display text encoding to <encoding>

The markup language is sent as plain text strings through JSON, which will be displayed directly. Special commands can be embedded inside the square bracket '[' ']' sets.
Escaping is supported for characters which might otherwise not be possible to insert. The characters '', '[' and ']' can be escaped with the sequences "\\", "\[" and "\]" respectively.
Hex data sequences can be inserted with the escape "\xHH", where "HH" is a double digit hex value.

History

Rev 1.0	29th September 2016	Official First Release
Rev 1.1	1st November 2016	Mistyping corrected
Rev 1.2	23 January 2017	• Update for firmware 1.2 device compatibility • Add example JSON data • Describe alternative job completion method (http GET) • Note compatibility issue for “text/plain” data
Rev 1.3	20 September 2017	• Describe a buzzer control method in GET response
Rev 1.4	9 March 2018	• Add description about peripheral device. • Add how to control of peripheral devices in JSON request/response (POST) • Describe a cut/dithering control method in GET response

Star Micronics is a global leader in the manufacturing of small printers. We apply over 50 years of knowhow and innovation to provide elite printing solutions that are rich in stellar reliability and industry-respected features. Offering a diverse line of Thermal, Hybrid, Mobile, Kiosk and Impact Dot Matrix printers, we are obsessed with exceeding the demands of our valued customers every day.

We have a long history of implementations into Retail, Point of Sale, Hospitality, Restaurants and Kitchens, Kiosks and Digital Signage, Gaming and Lottery, ATMs, Ticketing, Labeling, Salons and Spas, Banking and Credit Unions, Medical, Law Enforcement, Payment Processing, and more!

High Quality POS Receipts, Interactive Coupons with Triggers, Logo Printing for Branding, Advanced Drivers for Windows, Mac and Linux, Complete SDK Packages, Android, iOS, Blackberry Printing Support, OPOS, JavaPOS, POS for .NET, Eco-Friendly Paper and Power Savings with Reporting Utility, ENERGY STAR, MSR Reading, *future*PRNT, StarPRNT… How can Star help you fulfill the needs of your application?

Don't just settle on hardware that won't work as hard as you do. Demand everything from your printer. Demand a Star!

Star Micronics Worldwide

Star Micronics Co., Ltd. 536 Nanatsushinya
Shimizu-ku, Shizuoka 424-0066 Japan
+81-54-347-2163
http://www.star-m.jp/eng/index.htm

Star Micronics America, Inc.
65 Clyde Road. Suite G
Somerset, NJ 08873 USA 1-848-216-3300
http://www.starmicronics.com

Star Micronics EMEA
Star House
Peregrine Business Park, Gomm Road
High Wycombe, Buckinghamshire HP13 7DL
UK +44-(0)-1494-471111
http://www.star-emea.com

Star Micronics Southeast Asia Co., Ltd.
Room 2902C. 29th Fl. United Center Bldg.
323 Silom Road, Silom Bangrak, Bangkok 10500
Thailand
+66-2-631-1161 x 2
http://www.starmicronics.co.th/