

TechNote: Productionserver API Developers Guide

History:

- 2018-10-15 / Sebastian Wegner: Rev 1
- 2018-10-30 / Sebastian Wegner: Rev 2
 - Rearranged Chapter 4
 - GetJobStatus: Chapter 4.4.1
 - InsertOnTop, Abort job: Chapter 4.4.5
- 2018-11-28 / Sebastian Wegner: Rev 3
 - Responses in XML according to Accept header, Chapter 1.4.2
 - Starting the server, Chapter 1.5
 - Color prediction, Chapter 3.8
 - Get cost calculation data, Chapter 4.4.3
 - Get notifications, Chapter 4.4.5
 - Abort job, Chapter 4.5
 - Notification subscriptions, Chapter 6
- 2018-12-07 / Sebastian Wegner: Rev 4
 - New error notifications, Chapter 6.4
- 2019-01-21 / Jens Kopp: Rev 5
 - New job status and notifications for failures, Chapter 4.4.1 and 6.4
- 2019-04-05 / Sebastian Wegner: Rev 6
 - New endpoint “profiles”, Chapter 7
 - Color prediction: new parameter “inkSaving”, Chapter 3.8
 - Job settings: Breaking change. New section “color” for color management settings.
 - New parameter “inkSaving”. Chapter 4.5.1.1
- 2019-05-24 / Sebastian Wegner: Rev 7
 - New endpoint “systemLog”, Chapter 2.2
 - Changed endpoint infos to infos/status, Chapter 2.12.1
- 2019-08-07 / Sebastian Wegner: Rev 8
 - New job settings parameters, Chapter 4.5.1.3
- 2019-10-07 / Sebastian Wegner: Rev 9
 - Draft for linearization API, Chapter 7.
- 2019-11-11 / Sebastian Wegner: Rev 10
 - Reworked linearization API draft, Chapter 7, step now last part of the URL.
 - New endpoint “Get data”
- 2020-02-14 / Sebastian Wegner: Rev 11
 - Corrected droplet.count data type to Integer64, Chapter 4.4.3
 - Added option “redundantPatches” for target creation, Chapter 7.7
 - Renamed type Int and Integer to Integer32 to distinguish from Integer64
 - Re-added lost chapter of workflow settings, Chapter 4.5.1.7
- 2020-08-12 / Sebastian Wegner: Rev 12
 - Added GET list of notification subscriptions. Chapter 6.1
- 2020-08-13 / Sebastian Wegner: Rev 13
 - Added options for target printing. Chapter 7.8
- 2020-09-04 / Sebastian Wegner: Rev 14
 - Rip settings for antialiasing updated. Chapter 4.5.1.5
 - Added inkSavingTolerance. Chapter 4.5.1.1
- 2020-09-09 / Sebastian Wegner: Rev 15
 - Added colorCorrectionSettings. Chapter 4.5.1.2

- 2020-10-20 / Sebastian Wegner: Rev 16
Added result when printing targets. Chapter 7.8
- 2020-11-06 / Sebastian Wegner: Rev 17
Added Json result data for measurement data. Chapter 7.10
Renamed endpoint infos to system and added restart endpoint. Chapter 1.5.3
Added progressPercent and lastError to job status. Chapter 4.4.1
- 2021-01-12 / Sebastian Wegner: Rev 18
Added option to set patchData. Chapter 7.9
- 2021-02-23 / Sebastian Wegner: Rev 19
Changed blackpoint values field to enable multicolor. Chapter 7.11
Job settings will now be changed via endpoint "settings" (4.5.1) instead of a modifying action (4.5.2).
Added profile settings to color management settings. Chapter 4.5.1.1.
Added endpoints to access the color table of a job (4.4.6 and 4.5.2).
Added endpoint to access color table files. Chapter 8
Added endpoint to access MIMs. Chapter 3.9.
Remove MIM list from endpoint configuration (Chapter 3.7)
Added endpoint to access output profiles and linearizations. Chapter 3.10
- 2021-06-07 / Sebastian Wegner: Rev 20
Added workflowType and user defined media sizes to hotfolder list, Chapter 3.7
- 2021-06-30 / Sebastian Wegner: Rev 21
Added "CustomColorModel" for device color replacement, Chapter 8
- 2021-07-21 / Sebastian Wegner: Rev 22
Added "JDF" for hotfolder info, Chapter 3.7
- 2021-09-28 / Sebastian Wegner: Rev 23
Added ink splitting options for each channel, Chapter 7.9
- 2021-11-03 / Jens Kopp: Rev 24
Added endpoint „colorCorrection" to access the functionality of the Color Correction Loop Module (CCLM), Chapter 9
- 2021-11-18 / Sebastian Wegner: Rev 25
Added endpoints for additional jobs in „colorCorrection"
Chapters 9.3, 9.8, 9.9
- 2021-12-01 / Sebastian Wegner: Rev 26
Added target options and corrected description of „profiling GET data"
Chapter 7.10
- 2022-02-25 / Sebastian Wegner: Rev 27
Added setting and retrieving droplet separation curves Chapter 0 and 7.5
- 2022-03-09 / Sebastian Wegner: Rev 28
Changed output of Lch values to array, corrected data types of color data
Chapters 7.10
- 2022-05-19 / Sebastian Wegner: Rev 29
Brightness and contrast changed from integer to double type, Chapter 4.5.1.2
Added cropping parameters and output mirroring, Chapter 4.5.1.3
Added download of target files, Chapter 7.8
Added download of print output files, Chapters 4.4.1 and 4.6
- 2022-06-24 / Sebastian Wegner: Rev 30
Added endpoint "GET job settings", Chapter 4.4.7
Added options for control wedge target in job Settings, Chapter 4.5.1.3
Added endpoint "controlWedge", Chapter 10
Added endpoint "measurementDevices", Chapter 2.5
- 2022-07-06 / Sebastian Wegner: Rev 31
Added security hints, Chapter 1.5.3
Added control wedge workflow example, Chapter 10.5
Added new profiling options, Chapter 7.11

- 2022-07-14 / Sebastian Wegner: Rev 32
Changed data type of active and combinable channels to array, Chapter 7.11
- 2022-08-22 / Sebastian Wegner: Rev 33
Added support for Epson SpectroProofer, Chapters 7.3 and 7.8
- 2022-09-07 / Sebastian Wegner: Rev 34
Corrected value “maxDensityTargetValue” to “maxDensity”, Chapters 7.9 and 7.10
- 2022-10-05 / Sebastian Wegner: Rev 35
Added “maxDensityIndex” as alternative to “maxDensity”, Chapters 7.9 and 7.10
- 2022-11-08 / Sebastian Wegner: Rev 36
Added new endpoint for containers, Chapter 11
- 2022-12-12 / Sebastian Wegner: Rev 37
Added new fileInfos object containing output file information, Chapter 4.4.1
- 2022-12-13 / Sebastian Wegner: Rev 38
Added separation curves as read only parameter of profiling options, Chapter 7.11
- 2023-01-13 / Sebastian Wegner: Rev 39
Made option “downloadOutputFiles” available when creating a new job
with autoRip and autoPrint, Chapter 4.2
- 2023-01-18 / Sebastian Wegner: Rev 40
Output file information now also appear when “downloadOutputFiles” option is not
active, Chapter 4.4.1
- 2023-02-20 / Sebastian Wegner: Rev 41
Allow opening a profile in a specific queue, Chapter 7.12

Inhalt

1	Basic information	8
1.1	Introduction	8
1.2	REST	8
1.2.1	HTTP based	8
1.2.2	Versioning	8
1.2.3	Resources / Endpoints	9
1.2.4	Actions / Methods	9
1.3	HTTP requests	10
1.3.1	Basic components	10
1.3.2	Authorization	10
1.3.3	Message body	11
1.4	Responses	12
1.4.1	Status code	12
1.4.2	Response data	13
1.5	Starting the server	16
1.5.1	The server log window	16
1.5.2	Configuration	17
1.5.3	Security	18
2	Endpoint „system“	19
2.1	Status	19
2.2	System log	19
2.3	Clear log	20
2.4	Restart	20
2.5	Measurement devices	20
3	Endpoint “queues”	21
3.1	List queues	21
3.2	Open queue	21
3.3	Close queue	21
3.4	Get status	22
3.5	Set status	22

3.6	List print jobs.....	23
3.7	Get configuration	24
3.8	Color prediction	25
3.9	MIMs	27
3.9.1	List MIMs	27
3.9.2	Get MIM settings.....	27
3.9.3	Change MIM settings	28
3.10	Get profiles	28
4	Endpoint „jobs“	29
4.1	List jobs	29
4.2	Create new or duplicate job	30
4.3	Delete job.....	31
4.4	Get job data	31
4.4.1	Get job status	31
4.4.2	Get log.....	33
4.4.3	Get cost calculation data	34
4.4.4	Get preview file	35
4.4.5	Get notifications	35
4.4.6	Get color table	36
4.4.7	Get job settings	36
4.5	Change job data.....	37
4.5.1	Job settings	37
4.5.2	Color table	50
4.6	Job actions	51
5	Endpoint „files“	52
5.1	Upload file	52
5.2	Download file	52
5.3	Delete file.....	52
5.4	List files.	53
5.5	Get file information	53
6	Endpoint „notificationSubscriptions“	54
6.1	List subscriptions.....	54
6.2	Create subscription	55

6.3	Remove subscription.....	55
6.4	Sent notifications	56
7	Endpoint „profiles“	59
7.1	Create profile.....	60
7.2	Close	61
7.3	Get status	61
7.4	Get curves.....	63
7.5	Send curves	64
7.6	List targets	65
7.7	Create target	66
7.8	Print target	68
7.9	Send data.....	69
7.10	Get data	70
7.11	Calculate linearization / profile	72
7.12	Open profile	76
7.13	Workflow examples	76
8	Endpoint „colorTables“	79
8.1	List color tables	79
8.2	Get color table	79
8.3	Create color table entries	82
8.4	Change color table entries.....	82
8.5	Delete color table / entries	82
9	Endpoint „colorCorrections“.....	83
9.1	Create color correction	83
9.2	Delete color correction	83
9.3	Get color correction info.....	84
9.4	Create inspection system files	84
9.5	Create iteration	85
9.6	Delete iteration	85
9.7	Get status	86
9.8	Add jobs to correction.....	86
9.9	Remove jobs from correction	86

9.10 Workflow example	87
10 Endpoint „controlWedge“	88
10.1 Get targets	88
10.2 Get evaluation	88
10.3 Create evaluation	90
10.4 Delete evaluation	91
10.5 Workflow example	91
11 Endpoint „container“	92
11.1 Create container	92
11.2 Get container data	92
11.2.1 Get job list	92
11.2.2 Get settings	93
11.3 Change container data	96
11.3.1 Add jobs	96
11.3.2 Change settings	96
11.3.3 Change positions	96
11.3.4 Trigger auto arrangement	97
11.4 Remove container data	97
11.4.1 Remove jobs	97
11.4.2 Split container	97
12 Testing with Postman	98
12.1 Introduction	98
12.2 Example: Printing a file:	101

1 Basic information

1.1 Introduction

The Productionserver API provides access to basic functions of the Productionserver (PS) such as opening and closing printer queues or creating and tracking print jobs. The main goal is to embed PS into a workflow which is controlled by third party applications. This manual is intended for developers who want to implement a connection using it. The API is designed as a REST service. This means it is providing a HTTP server which is accepting requests on a specific TCP port. Once the server is started you can send HTTP requests to the application and receive responses from it. Readers should be familiar with basic HTTP communication to understand this manual.

Because HTTP is the same protocol internet browsers use to access web pages it is possible to use a browser to create requests and test the functionality of the API. There is also useful software such as [Postman](#) which makes it easy to create HTTP requests and show the results of the server. A collection of Postman requests accompanies this manual, so you can test the basic functionality before implementing your own client. See chapter 6.

1.2 REST

[REST](#) (Representational state transfer) is a design model for communication between applications in a network. It is mostly based on the HTTP protocol and defines some basic principles which the Productionserver tries to follow whenever appropriate:

1.2.1 HTTP based

The server is available in the network and responds to requests on a TCP port. By default, the server uses secure communication over HTTPS and listens on port 443. Assuming you have a computer with the name “MyProductionserver” the service will be available with an URI like that:

<https://MyProductionserver:443>

1.2.2 Versioning

As the implementation of the API may evolve it is necessary to define the version which should be used. Future implementations could change the API in a way that existing calls are not compatible anymore. To avoid this, you must always provide the version as the first element in a request URI:

<https://MyProductionserver:443/v1>

The current version of the API is 1, so appending v1 is the only option.

1.2.3 Resources / Endpoints

A server provides a resource via an address. This address is also called an endpoint. Sub-addresses are used if a single instance or resource should be accessed.

Access the list of printer queues:

<https://MyProductionserver:443/v1/queues>

Access a single printer queue with the name “MyQueue”:

<https://MyProductionserver:443/v1/queues/MyQueue>

Get the job list of the printer queue “MyQueue”:

<https://MyProductionserver:443/v1/queues/MyQueue/jobs>

1.2.4 Actions / Methods

The basic actions which should be done with the resource are expressed as HTTP methods.

The following standard HTTP methods will be used:

- GET requests ask for information without modifying the resource.
- PUT requests modify an existing resource.
- POST requests create a new resource.
- DELETE requests delete an existing resource.

1.3 HTTP requests

1.3.1 Basic components

A complete HTTP request consists of four basic elements:

- The URI to access the resource on a server.
- The method to express the desired action.
- The message header to provide additional information such as the type of the message body and authorization data.
- The message body which contains additional data such as parameters or binary file data.

This documentation describes a request in the following form:

GET /v1/endpointName/resourceName			Method and URI
Parameters:			Name and type of parameters
param1	Bool	Description of param1	
param2	Integer32	Description of param2	
Result:			Name and type of results
result1	String	Description of result1	
result2	Integer32	Description of result2	

The message header will not be documented explicitly because it will be filled with HTTP standard elements. You always must provide the authorization credentials and the content-type field. Optionally you can provide an accept-language field to select the preferred language for localized content (for example “de” for German).

1.3.2 Authorization

Each request must contain username and password of an existing user in the Access Control Module (ACM).

The server uses the Basic Authentication scheme.

1.3.3 Message body

The message body can be JSON data, XML data or any binary data. If a request needs parameters JSON is the preferred way to send them. A raw request can look like this:

```
GET https://MyRESTServer:443/v1/endpointName/resourceName
```

```
content-type:"application/json"  
cache-control:"no-cache"  
authorization:"Basic V2VnbmVyOmJvY2t3dXJzdDIx"  
accept-language:"de"
```

```
{  
  "param1": true,  
  "param2": 12  
}
```

The same request can also use XML instead of JSON. The name of the root element will be ignored by the server and can be any name you choose.

```
GET https://MyRESTServer:443/v1/endpointName/resourceName
```

```
content-type:"text/xml"  
cache-control:"no-cache"  
authorization:"Basic V2VnbmVyOmJvY2t3dXJzdDIx"  
accept-language:"de"
```

```
<?xml version="1.0" encoding='UTF-8'?>  
<root>  
  <param1>true</param1>  
  <param2>12</param2>  
</root>
```

Parameters can also be appended to the URI instead:

<https://MyRESTServer:443/v1/endpointName/resourceName?param1=true¶m2=12>

1.4 Responses

1.4.1 Status code

Each response will at least provide a HTTP status code as a general type of feedback. A complete list of defined status codes can be found [here](#). A successful request will usually be answered with HTTP_OK = 200, a successful POST request with HTTP_CREATED = 201.

The following status codes are used by the server:

Code	Name	Description
200	Ok	Successful GET- PUT- or DELETE request.
201	Created	Successful POST-request.
400	Bad request	Invalid parameters.
401	Unauthorized	User or password invalid or authorization not possible.
403	Forbidden	The resource cannot be accessed because it is usable only for another user/process.
404	Not found	The resource was not found.
405	Method not allowed	The specified method is not allowed by this endpoint.
409	Conflict	The resource is used by another user/process.
422	Unprocessable entity	The requested action cannot be performed.
429	Too many requests	Too many requests at once.
500	Internal server error	Mostly because of programming errors.
501	Not implemented	Endpoint exists but not implemented yet.
503	Service unavailable	Server is in maintenance mode.

1.4.2 Response data

Response data is usually in JSON if the response is text data. Check the Content-Type field of the response message header. If the request provides an Accept message header and it is set to text/xml or application/xml the server will respond in XML instead of JSON.

Most text responses will provide status information about the handled request:

- user: The name of the user who created the request.
- version: The version of the request.
- endpoint: The accessed endpoint.
method: The method of the request.
- code: The HTTP result code which is also part of the common HTTP result.
- text: The standard text of the HTTP result.
- time: The time needed to handle the request in milliseconds.
- error: Optional additional information about the cause of an unsuccessful operation.

A successful request contains the status block at the beginning followed by the payload data:

```
{
  "status": {
    "user": "Wegner",
    "version": "v1",
    "endpoint": "queues",
    "method": "GET",
    "code": 200,
    "text": "OK",
    "time": 128
  },
  "queueName": "PDF FLAT HIGH-RES",
  "printerName": "PDF FLAT HIGH-RES",
  "printerID": 362,
  "printerCaps": {
    "hasRoll": true,
    "hasCutter": false,
    "supportsBorderlessPrinting": false
  },
  "hotfolders": [
    {
      "name": "PDF FLAT HIGH-RES",
      "path": "C:\\ProgramData\\ColorGATE Software\\Production-
server10\\HotDir\\PDF FLAT HIGH-RES\\",
      "active": true
    }
  ],
  "media": [
    {
      "name": "Paper",
```

```
"ID": "C1B731C1",
"type": "Private",
"Inks": [
  {
    "name": "Ink",
    "ID": "00014522",
    "type": "Private",
    "MetaModes": [
      {
        "name": "Mode",
        "ID": "002C1EA5",
        "type": "Private"
      }
    ]
  }
],
"mediaSizes": [
  {
    "name": "Roll - ISO A3",
    "ID": "00002711",
    "width": 297,
    "height": 60000
  },
  {
    "name": "Roll - ISO A2",
    "ID": "00002712",
    "width": 420,
    "height": 60000
  }
]
```

If you are only interested in special parts of the response you can provide the parameter `responseFields=value1,value2,..` with the name of the fields you are interested in, for example

<https://MyRESTServer:443/v1/queues/queueName?responseFields=queueName,hotfolders>.

This only applies to the root elements of the json (or XML) data. The status information will always be delivered completely.

```
{
  "status": {
    "user": "Wegner",
    "version": "v1",
    "endpoint": "queues",
    "method": "GET",
    "code": 200,
    "text": "OK",
    "time": 182
  },
  "queueName": "PDF FLAT HIGH-RES",
  "hotfolders": [
    {
      "name": "PDF FLAT HIGH-RES",
      "path": "C:\\ProgramData\\ColorGATE Software\\Production-
server10\\HotDir\\PDF FLAT HIGH-RES\\",
      "active": true
    }
  ]
}
```

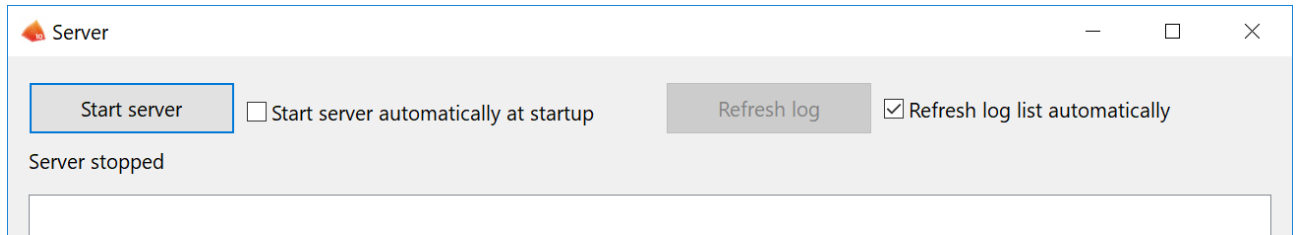
An unsuccessful request only contains the status block. The error field contains additional information about the cause of failure:

```
{
  "status": {
    "user": "Wegner",
    "version": "v1",
    "endpoint": "queues",
    "method": "put",
    "code": 400,
    "text": "Bad request"
    "time": 954,
    "error": "Resource name missing"
  }
}
```

1.5 Starting the server

1.5.1 The server log window

To start the server manually the menu command Options->REST Server must be chosen. This will open the server log window. All other windows are now disabled to suppress access of the GUI while remote requests are served.



The button “Start server” can be pressed to start the server. The button text will change to “Stop server”, so the server can be stopped by pressing it again. If the log window gets closed by the user, the server will not stop but turn into maintenance mode. This way remote requests will be rejected with a 503 HTTP_SERVICE_UNAVAILABLE status.

The option “Start server automatically at startup” can be used to start the server immediately after opening the program. This way it is not necessary anymore to start it manually.

The server automatically logs any incoming request which can be viewed in the log list. The button “Refresh log” can be used to reload the current log file and display it in the list. The option “Refresh list automatically” can be used to reload the current log file once per second. This can be turned off for performance reasons.

1.5.2 Configuration

There is no GUI to configure the server's parameters. Any option differing from the default configuration must be configured in the ini file. For this purpose a [REST_API] section must be created if not already present.

```
APPE4.8Dir=C:\Sources\Mainline\Installatio
ACDBTest_State=2
[REST_API]
AutoStartServer=0
AutoRefreshLog=1
Port=449
[Settings]
hplfpsdkloglevel=verbose
```

The following parameters can be set:

Name	Description
AllowGUI	0 or 1, by default the GUI is disabled when the server is running. This behavior can be turned off with this switch. However, this is not recommended in a production workflow because program stability cannot be guaranteed when using the GUI at the same time when remote requests are served. Default is 0.
AutoRefreshLog	Controls if the log list will be refreshed automatically. Can be configured in the log window. Default is 0.
AutoStartServer	Controls if the server starts automatically after program startup. Can be configured in the log window. Default is 0.
MaxQueuedRequests	Number of simultaneously accepted requests. If a request cannot be served immediately (see the following parameter) it will be queued and served later. If this number is exceeded the request will be rejected. Default is 1000.
MaxThreads	Number of simultaneously served requests. Default is 16.
Port	The port over which the server is accessible. If you don't provide this parameter the default ports are used (80 for unsecure connection over http, 443 for secure connection over https).
UseSSL	Controls if secure connections shall be used. 0 = Unsecure access via http 1 = Secure access via https Default is 1. It is not recommended to use an unsecure connection.
LogJsonBody	Logs every single request in a file in CGLogs. Useful for support cases. Should not be enabled generally because it will produce many files and decrease performance. 0 = Disabled (Default) 1 = Enabled

1.5.3 Security

1.5.3.1 Certificate

By default, all connections are secured by the https protocol. For this purpose, a self-signed certificate is used for authorization. You find the certificate files “certificate.pem” and “key.pem” in the installation directory:

C:\Program Files (x86)\ColorGATE Software\ProductionserverXX\SSL

If you want to use your own certificate you can copy your own files to this location. From V22.10 on it is possible to put your files into the program data folder. This is the recommended way because the files are not overwritten when installing a Productionserver update. As long as the filenames are the same Productionserver automatically uses them, if they are present.

C:\ProgramData\ColorGATE Software\ProductionserverXX\SSL

1.5.3.2 TLS

From V22.10 the minimum required TLS version is 1.2, in older version it was 1.0. Because older TLS versions than 1.2 have numerous known security issues it is not recommended to use them. However, if needed you can change the minimum required TLS version with an ini setting:

```
[Settings]
MinimumTLSVersion=1.1
```

Possible values are 1.0, 1.1, 1.2 and 1.3

2 Endpoint „system“

This endpoint accesses general system data.

2.1 Status

Checks if the server is available and if the user is authorized to access the API. As a result, you retrieve basic server information.

GET system/status		
Result:		
product	String	Product name.
version	String	Version number.
serialNumber	String	Serial number.
serverStart	String	Server start in UTC time.
serverUptime	String	Time since server has started.
versionAPI	String	Maximum supported version.

2.2 System log

Use this endpoint to retrieve the system log.

GET system/log		
Result:		
log	Array	Log messages.

Array log		
event	String	The event. See the chapter 0 for possible values.
date	String	Date of the event in the format YYYY-MM-DD.
time	String	Time of the event in the format HH:MM:SS.
queueName	String	Optional: The name of the queue. This will only be sent for the notification identifiers "Queue.XXX".
jobID	String	Optional: The identifier of the job. This will only be sent for the notification identifiers "Job.XXX".
fileName	String	Optional: The filename of the job. This will only be sent for the notification identifiers "Job.XXX".
data	Array	Optional: A list of textual data. It is present in the notifications Job.ContainerAdd, Job.ContainerRemove and Job.PrintPageFinished, Job.RipGeneralFailure and Job.PrintGeneralFailure.

2.3 Clear log

Use this endpoint to clear the system log. Returns the list of removed log entries. This way it is possible to retrieve and clear the system log in one step.

DELETE system/log		
Result:		
log	Array	Log messages.

2.4 Restart

It can be useful to restart the application regularly to avoid problems caused by memory leaks or other issues.

Use this endpoint to force a restart of the Productionserver application. After restart, the REST service will start automatically.

If no other requests are currently being served and no jobs are currently processing the application immediately restarts and the server is no longer responding. You can then poll the status (Chapter 2.1) to check if the server is running again.

If other requests are currently being processed a HTTP_FORBIDDEN result will be returned.

If any job is currently processing, you will also receive a HTTP_FORBIDDEN result and a list of jobs, which are currently processing.

GET system/log		
Result:		
jobs	Array	Optional: Contains list of currently processing jobs.

2.5 Measurement devices

Gets a list of all licensed measurement devices. The name of the device is used in various other endpoints.

GET system/measurementDevices		
Result:		
measurementDevices	Array	The list of devices

Array measurmentDevices		
name	String	The name of the device. This is used as the identifier of a device in various endpoints.
descriptiveName	String	Optional: The name of the device for display if it differs from the name.

3 Endpoint “queues”

This endpoint accesses open printer queues of the Productionserver. Queues are identified by their name, so the URI path must have the “queueName” segment appended to access a single queue.

3.1 List queues

Gets a list of open printer queues:

GET queues		
Result:		
queues	Array	The list of queues.

Array queues		
queueName	String	The name of the queue.
device	String	The name of the printer.

3.2 Open queue

Opens a printer queue. The queue itself will not be created (as POST may suggest) but an existing cos file will be opened. This way a new accessible entity of a queue is created.

POST queues		
Parameters:		
fileName	String	Name of the cos file to load.
Result:		
queueName	String	The name of the queue. This must be used in all following requests to access the queue.

3.3 Close queue

Closes a printer queue. The queue itself will not be deleted (as DELETE may suggest) but the cos file will be closed. Any changes (made manually before starting REST) will be saved. You can reopen it with POST whenever needed.

DELETE queues/queueName		
Result:		
n/a		

3.4 Get status

Gets the status of a queue:

GET queues/queueName/status		
Result:		
ripEnabled	Bool	Status of the rip queue.
printEnabled	Bool	Status of the printer queue.

3.5 Set status

Sets the status of a queue:

PUT queues/queueName/status		
Parameters:		
ripEnabled	Bool	Status of the rip queue.
printEnabled	Bool	Status of the printer queue.
Result:		
n/a		

3.6 List print jobs

Gets a list of all the jobs in the queue:

GET queues/queueName/jobs		
Result:		
jobs	Array	The list of jobs.

Array jobs		
jobID	String	The identifier of the job.
jobName	String	The name of the job.
fileName	String	The name of the file to print.
size	String	The size of the job.
copies	Integer32	The number of copies.
creationDate	String	The creation date of the job.
fileSize	String	Optional: File size of the job.
ripped	Bool	If the job was ripped.
printed	Bool	If the job was printed.
printDate	String	Optional: The print date of the job.
backup	Bool	If the job was backed up.
backupDate	String	Optional: The backup date of the job.
preview	Bool	If preview data exists.
costCalc	Bool	If cost calculation data exists.
container	Bool	If the job is a container.

3.7 Get configuration

Gets the configuration of the queue:

GET queues/queueName/config		
Result:		
queueName	String	Identifier of the queue.
printerName	String	Name of the printer.
printerID	String	Identifier of the printer.
printerCaps	Object	Printer capabilities.
hotfolders	Array	Optional: The list of hotfolders.
mediaSizes	Array	Optional: The list of media.

Object printerCaps		
hasRoll	Bool	
hasCutter	Bool	
supportsBorderlessPrinting	Bool	

Array hotfolders		
name	String	The name you must provide when creating a new job.
path	String	The internal path of the hotfolder.
active	Bool	
workflowType	String	Name of the workflow type. “Production”, “Proof”, “Screen”
mediaSizesUserDefined	Array	List of user defined media
JDF	Bool	True = The hotfolder is a JDF folder and cannot be used for REST.

Array mediaSizes, mediaSizesUserDefined		
name	String	The name of the media
ID	String	The internal ID of the media
width, height	double	Media size
marginLeft, marginRight, marginTop, marginBottom	double	Optional: margins, only if > 0.0

3.8 Color prediction

Calculates spot color estimation data. Either a MIM combination must be provided as parameter or the hotfolder whose current MIM combination should be used for calculation.

PUT queues/queueName/colorPrediction		
Parameters:		
mim	Object	Optional: Color management settings.
hotfolder	String	Optional: Name of the hotfolder which settings should be used. Only needed if no mim is set.
colors	Array	List of colors for which the calculation shall be processed.
inkSaving	String	The following values are supported: "None" "Min" "Medium" "Max"

Object mim		
media	String	Media name of requested MIM-combination.
ink	String	Ink name of requested MIM-combination.
metaMode	String	Metamode name of requested MIM-combination.

Array colors		
name	String	Identifier for the color.
type	String	Type of the color values. Possible values: "LAB", "CMYK" "RGB" "GRAY"
values	Array	List of double values. Depending of the type the length of this list varies.

Result:

Result		
colors	Array	List of colors

Array colors		
name	String	Identifier for the color.
inputColor	Object	A copy of the color data of the request.
ok	Bool	Identifies if the calculation was successful.
predictedColor	Object	The resulting Lab color. Only available if the calculation was successful.
deltaE76	Double	The delta E of the predicted color. Only available if the calculation was successful.
deltaE94	Double	The delta E of the predicted color. Only available if the calculation was successful.
deltaE00	Double	The delta E of the predicted color. Only available if the calculation was successful.

Object inputColor		
name	String	Identifier for the color.
type	String	Type of the color values. Possible values: “LAB”, “CMYK” “RGB” “GRAY”
values	Array	List of double values. Depending of the type the length of this list varies.

Object predictedColor		
type	String	Type of the color values. Always “LAB”
values	Array	Three Lab values.

3.9 MIMs

3.9.1 List MIMs

Gets a list of all the MIMs available in this queue.

GET queues/queueName/mim		
Result:		
mims	Array	Array with all MIM combinations.

Array mims		
media	String	Media name of MIM-combination.
ink	String	Ink name of MIM-combination.
metaMode	String	Metamode name of MIM-combination.
colorMode	String	Name of the color mode. "CMYK", "RGB"...
workflowType	String	Name of the workflow type. "Production", "Proof", "Screen"

3.9.2 Get MIM settings

Gets the settings of a particular MIM.

GET queues/queueName/mim		
Parameters:		
mim	Object	The selected MIM combination.
Result:		
mimSettings	Object	The settings of the selected MIM combination.

Object mim		
media	String	Media name of requested MIM-combination.
ink	String	Ink name of requested MIM-combination.
metaMode	String	Metamode name of requested MIM-combination.

Object mimSettings		
media	String	Media name of MIM-combination.
ink	String	Ink name of MIM-combination.
metaMode	String	Metamode name of MIM-combination.
profileSettings	Object	Settings from the "Profiles" tab of the advanced settings dialog. See chapter 4.5.1.1

3.9.3 Change MIM settings

Changes the settings of a particular MIM.

PUT queues/queueName/mim		
Parameters:		
mimSettings	Object	The settings of the MIM combination to change. See chapter 3.9.2

3.10 Get profiles

Gets a list of the available output profiles and linearizations.

GET queues/queueName/profiles		
Result:		
outputProfiles	Array	The list of available output profiles.
outputLinearizations	Array	The list of available output profiles.

Array outputProfiles		
name	String	Name of the profile.
colorMode	Array	Color mode of the profile. Not available for most generic color modes (4CLR and more).
countChannels	Integer32	The number of channels in the color mode.

Array outputLinearizations		
name	String	Name of the linearization.
colorMode	Array	Color mode of the linearization.

4 Endpoint „jobs“

This endpoint maintains print jobs. New jobs can be created, existing jobs can be tracked, modified or deleted.

4.1 List jobs

Gets a list of all jobs in all printer queues.

GET jobs		
Result:		
jobs	Array	The list of all jobs in all queues.

Array jobs		
queueName	String	The name of the queue the job is in.
jobID	String	The identifier of the job.
jobName	String	The name of the job.
jobStatus	String	The status of the job: The following values are possible: “Idle”, “Printing”, “Ripping”, “Copying” “Capturing usage data” “Creating preview”, “Creating backup”, “Printing failed”, “Ripping failed”, “Copying failed”, “Analysis failed”, “Errors pending”
progressPercent	Integer32	Optional: Progress between 0 and 100 if status is “Printing”, “Ripping” or “Creating preview” or “Creating backup”.
lastError	String	Optional: Last error message of the job log if status is “Printing failed”, “Ripping failed”, “Copying failed”, “Analysis failed” or “Errors pending”. The error has the same format like one entry of the job log in chapter 0.
fileName	String	The name of the file to print.
size	String	The size of the job.
copies	Integer32	The number of copies.
creationDate	String	The creation date of the job.
fileSize	String	Optional: File size of the job.
ripped	Bool	If the job was ripped.
printed	Bool	If the job was printed.
printDate	String	Optional: The print date of the job.
backup	Bool	If the job was backed up.

backupDate	String	Optional: The backup date of the job.
preview	Bool	If preview data is ok.
costCalc	Bool	If cost calculation data is ok.
colorCorrection	Bool	Optional: If color correction loop was applied.
container	Bool	If job is a container.

4.2 Create new or duplicate job

Creates a print job. If you create a new one the associated file must be uploaded to the “files” endpoint first.

You can also duplicate an existing job here.

POST jobs		
Parameters:		
queueName	String	The name of the queue where the job shall be created.
hotfolder	String	Name of the hotfolder which settings should be applied. Only needed when creating a new job.
fileID	Integer32	ID of the associated file. You get this number when uploading the file to the “files” endpoint. Only needed when creating a new job.
duplicateJobID	String	Optional: The ID of the job you want to duplicate. If you provide this parameter, “hotfolder” and “fileID” are not needed and will be ignored.
settings	Object	Job settings, which override the hotfolder settings. See chapter 4.5.1 for available options.
downloadOutputFiles	Bool	Optional: When creating a new job and automatic ripping and printing is enabled a value of true makes the output files for the printer available via REST. This is only useful for file-based printers. The files can be downloaded with the “Files” endpoint. You can obtain the file IDs via job status endpoint 4.4.1 or the notification Job.PrintPageFinished.
Result:		
		See contents of Array jobs in chapter 4.1.

4.3 Delete job

Removes the job.

DELETE jobs/jobID		
Result:		
n/a		

4.4 Get job data

These endpoints provide information and data for a particular job.

4.4.1 Get job status

Get the status of the job.

GET jobs/jobID/status		
Result:		
queueName	String	The name of the queue the job is in.
jobID	String	The identifier of the job.
jobName	String	The name of the job.
jobStatus	String	The status of the job: The following values are possible: “Idle”, “Printing”, “Ripping”, “Copying” “Capturing usage data” “Creating preview”, “Creating backup”, “Printing failed”, “Ripping failed”, “Copying failed”, “Analysis failed”, “Errors pending”
progressPercent	Integer32	Optional: Progress between 0 and 100 if status is “Printing”, “Ripping” or “Creating preview” or “Creating backup”.
lastError	String	Optional: Last error message of the job log if status is “Printing failed”, “Ripping failed”, “Copying failed”, “Analysis failed” or “Errors pending”. The error has the same format like one entry of the job log in chapter 0.
fileName	String	The name of the file to print.
size	String	The size of the job.
copies	Integer32	The number of copies.
creationDate	String	The creation date of the job.
fileSize	String	Optional: File size of the job.
ripped	Bool	If the job was ripped.

printed	Bool	If the job was printed.
printDate	String	Optional: The print date of the job.
backup	Bool	If the job was backed up.
backupDate	String	Optional: The backup date of the job.
preview	Bool	If preview data exists.
costCalc	Bool	If cost calculation data exists.
outputFiles	Array	Optional: Only available for file-based printers.

Array outputFiles		
fileID	Integer32	Optional: The ID of the file if a print action demands to download the output files (4.6).
filename	String	The name of the output file
fileType	String	The type of the output file. This value is driver dependent. For most file-based drivers it can either be "SeparationFile" or "PreviewFile".
fileInfos	Object	Optional, only if driver provides additional data for the output file. Content is driver dependent.

The fileInfo object is driver dependent. However, many drivers provide the following information about an output file.

Object fileInfo		
widthPixel	Integer32	The width of in pixels.
heightPixel	Integer32	The height in pixels.
resolutionX	Double	The horizontal resolution.
resolutionY	Double	The vertical resolution.
widthMM	Double	The width in mm.
heightMM	Double	The height in mm.

4.4.2 Get log

Gets the log messages of the current log file.

GET jobs/jobID/log		
Result:		
log	Array	The list of all log messages.

Array log		
severity	String	The following values are possible: “debug” “info” “warning” “error” “fatal”
time	String	The timestamp of the log message.
source	String	The application source of the log. “FRONTEND” “ANALYZE” “RIP” “PRINT”
text	Array	A string array with all log messages.

4.4.3 Get cost calculation data

Gets the current cost calculation data of the job.

GET jobs/jobID/cost		
Result:		
usageDataCalculated	Object	Calculated usage data generated when the job is printed, or when cost has been estimated before printing.
usageDataReported	Object	Optional: Reported usage data generated when the printer has completely finished printing the job.
costData	Object	Job cost data.

Object usageDataCalculated and usageDataReported		
ink	Double	Ink consumption, in milliliters.
printedMedia	Double	Amount of media that was printed on in square meters.
wastedMedia	Double	Amount of media that was produced but not printed on in square meters.
multiCopy	Integer32	Amount of copies of the original job.
channels	Array	Per channel information about ink usage and droplet counts.

Array channels		
ID	Integer32	Channel ID.
name	String	Name of the channel.
ink	Double	Ink consumption for the channel, in milliliters.
droplets	Array	Amount of copies of the original job (only available for usageDataCalculated).

Array droplets		
size	Integer32	Index for the droplet size.
amount	Double	Ink amount per droplet of the given size, in picoliters.
count	Integer64	Number of droplets of the given size.

Object costData		
total	Double	Total job cost.
ink	Double	Cost of ink used for the job.
media	Double	Cost of media used for the job.
additional	Double	User defined additional charges added to total job cost.
surcharge	Double	Percentage value which affects total job cost.

4.4.4 Get preview file

Downloads the current preview file of the job. If the request is successful, the result data will be an image file. Otherwise it is the JSON status data containing an error description. Check the HTTP status and the content-type field of the response.

GET jobs/jobID/preview		
Parameters:		
imageType	String	Optional: If you don't provide this parameter a TIFF file will be created. The following values are supported: "image/tiff" = Compressed TIFF format. "image/png" = PNG format.
Result:		
If HTTP status is HTTP_OK the Content-Type header field is set to "image/tiff" or "image/png". The message body contains the binary data of the image file. If an error occurred, the HTTP status is not HTTP_OK, the Content-Type header is set to "application/json" and the message body contains the status data.		

4.4.5 Get notifications

Many actions generate a notification which can be received by subscribers. See chapter 6 for more information.

This endpoint provides the complete history of all notifications of a job. Because it accesses the system log it should not be used for polling the job status. Instead use the much more efficient endpoint 4.4.1.

GET jobs/jobID/notifications		
Result:		
notifications	Array	

Array notifications		
notification	String	The notification identifier. See the chapter 0 for possible values.
date	String	Date of the notification in the format YYYY-MM-DD.
time	String	Time of the notification in the format HH:MM:SS.
jobID	String	Optional: The identifier of the job. This will only be sent for the notification identifiers "Job.XXX".
queueName	String	Optional: The name of the queue. This will only be sent for the notification identifiers "Queue.XXX".
data	Array	Optional: A list of textual data. It is present in the notifications Job.ContainerAdd, Job.ContainerRemove and Job.PrintPageFinished, Job.RipGeneralFailure and Job.PrintGeneralFailure.

Array data		
key	String	Identifier of the data value. Possible values are: When notification is Job.ContainerAdd or Job.ContainerRemove: "ContainerName": Name of the container job. When notification is Job.PrintPageFinished: "PageCount": Number of total pages to print. "PageNumber": Number of the current printed page. Specific printer drivers may add further data. See the driver documenta- tion. When notification is Job.RipGeneralFailure or Job.PrintGeneralFailure: "ErrorMsg": Description of the error.
value	String Integer32	Value.

4.4.6 Get color table

Gets the color table of the job.

GET jobs/jobID/colorTable		
Result:		
See chapter 8.2		

4.4.7 Get job settings

Retrieves all settings which can be set in endpoint 4.5.1

GET jobs/jobID		
Result:		
Settings	Object	See chapter 4.5.1 for the description of every field.

4.5 Change job data

4.5.1 Job settings

Applies job settings.

PUT jobs/jobID/settings		
Parameters:		
settings	Object	

The parameter “settings” can also be used when creating a new job (Chapter 4.2). Every parameter is optional.

The settings section is divided into eight sub-sections which are all optional:

Object settings		
color	Object	Color management settings.
job	Object	Job settings.
printer	Object	Printer settings.
rip	Object	RIP settings.
driver	Object	Printer driver specific settings.
workflow	Object	Workflow settings.
admin	Object	Job administrative values.
formData	Object	Field data for PDF forms.

A settings section could look like this:

```
"settings": {
  "color": {
    "mim": {
      "media": "Paper",
      "ink": "MyInk",
      "metaMode": "HD"
    }
  },
  "job": {
    "width": 275.0,
    "height": 163.0,
    "cutmarks": true
  },
  "driver": {
    "explicitWidth": 200
  }
  "workflow": {
    "autoPrint": "True"
  }
}
```

```
},  
"formData": {  
  "fields": [  
    "name",  
    "email"  
  ],  
  "sets": [  
    [ "Sebastian Wegner", "sebastian.wegner@colorgate.com" ],  
    [ "Jens Kopp", "jens.kopp@colorgate.com" ]  
  ]  
}
```

4.5.1.1 Color management settings

The settings described in this section are related to the Color Management tab of the hotfolder properties dialog.

Object color		
mim	Object	Mim settings.
inkSaving	String	The following values are supported: “None” “Min” “Medium” “Max”
inkSavingTolerance	Double	Applicable if “inkSaving” is not “None”, a value between 0 and 5.
profileSettings	Object	Settings from the “Profiles” tab of the advanced settings dialog.

A mim setting must always specify all three fields.

Object mim		
media	String	Media name of requested MIM-combination.
ink	String	Ink name of requested MIM-combination.
metaMode	String	Metamode name of requested MIM-combination.

Object profileSettings		
useInputProfilesRGB	Bool	Use RGB profiles for both vector objects and bitmap objects.
useInputProfilesRGBvector	Bool	Use RGB profiles for vector objects.
useInputProfilesRGBbitmap	Bool	Use RGB profiles for bitmap objects.
inputProfileRGB	String	Name of the RGB input profile.
inputRenderingIntentRGB	String	Rendering intent for the RGB input profile. Possible values: "Perceptual" "RelativeColorimetric", "AbsoluteColorimetric", "Saturation", "AbsoluteCompression", "LightenerCompensation", "MinimumWhiteCompression", "Absolute Perceptual", "BlackpointCompensation"
useInputProfilesCMYK	Bool	Use CMYK profiles for both vector objects and bitmap objects.
useInputProfilesCMYKvector	Bool	Use CMYK profiles for vector objects.
useInputProfilesCMYKbitmap	Bool	Use CMYK profiles for bitmap objects.
inputProfileCMYK	String	Name of the CMYK input profile.
inputRenderingIntentCMYK	String	Rendering intent for the CMYK input profile. Possible values: See inputRenderingIntentRGB.
useInputProfilesGray	Bool	Use Gray profiles for both vector objects and bitmap objects.
useInputProfilesGrayvector	Bool	Use Gray profiles for vector objects.
useInputProfilesGraybitmap	Bool	Use Gray profiles for bitmap objects.
inputProfileGray	String	Name of the Gray input profile.
inputRenderingIntentGray	String	Rendering intent for the Gray input profile. Possible values: See inputRenderingIntentRGB.
inputRenderingIntentLab	String	Rendering intent for the LAB input profile. Possible values: See inputRenderingIntentRGB.
useEmbeddedProfile	Bool	Use embedded profile.
usePdfRenderingIntents	Bool	Use PDFG rendering intents.

useBlackpointCompensation	Bool	Use black point compensation.
preservePureColors	String	Possible values: "None", "Black", "CMY", "CMYK", "CMYKRGB" (corresponds to C,M,Y,K, MY,CY,CM), "CMYRGB" (corresponds to C,M,Y, MY,CY,CM), "Custom"
preservePureColorExceptions	Array<Bool>	Only applicable if "preservePureColors" = "Custom". Possible values: "pureBlack", "purePrimaries", "pureCyan", "pureMagenta", "pureYellow", "pureSecondaries", "pureMY", "pureCY", "pureCM", "duplex", "triplex", "filterException"
useAdaptionProfile	Bool	Activates the adaption profile.
adaptionProfile	String	The name of the adaption profile.
useOutputProfile	Bool	Activates the output profile.
outputProfile	String	The name of the output profile.
useOutputLinearization	Bool	Activates the output linearization.
outputLinearization	String	The name of the output linearization.
useDeviceLinks	Bool	Activates the device link.

deviceLinkExceptions	Array<Bool>	<p>Only applicable if “useDeviceLinks” = true. Possible values:</p> <p>“pureBlack”, “blackOverprint”, “fullBlack”, “preserveZeroBlack”, “linearizeGray”, “pureGray”, “pureCMY”, “maxCMY”, “purePrimaries”, “pureCyan”, “pureMagenta”, “pureYellow”, “limitPrimaries”, “pureSecondaries”, “pureMY”, “pureCY”, “pureCM”, “pureRGB”, “maxRGB”, “pureWhite”, “duplex”, “triplex”, “purifyColor”, “filterException”, “clipToEdge”</p>
saturationEnhancement	String	<p>The following values are supported:</p> <p>“None” “Low” “Medium” “High” “Extreme”</p>

4.5.1.2 Color correction settings

The settings described in this section are related to the Color Correction tab of the Advanced color settings dialog.

Object colorCorrection		
brightness	Double	Value between -100 and 100.
contrast	Double	Value between -100 and 100.
gamma	Double	Value between 0 and 20.
gradationCurves	Array	Gradation curves.

Array gradationCurves		
channelName	String	Name of the channel which the curve is for. If it is empty the values apply to all channels. See chapter 0 for possible channel names.
curvePoints	Array	Array of at least two curve points.

Object curvePoints		
type	String	Type of the point: “Bezier” for a Bezier point smoothing the curve. “Corner” for a supporting point defining the curve.
x	Double	X coordinate of the point between 0 and 1.
y	Double	Y coordinate of the point between 0 and 1.

4.5.1.3 Job settings

The settings described in this section are related to the Job tab of the properties dialog.

Object job		
jobName	String	Name of the job.
width	Double	Width of a single copy of the job to be produced in mm. Note: If this parameter is set to zero, the width will be calculated by applying the aspect ratio of the image to the value given by parameter "height".
height	Double	Height of a single copy of the job to be produced in mm. Note: If this parameter is set to zero, the height will be calculated by applying the aspect ratio of the image to the value given by parameter "width".
scaleX scaleY	Double	Scaling factors. Note: If one of these parameters are specified, the attributes 'width' and 'height' are ignored.
rotation	String	The following values are supported: "Rotate0": Do not rotate. "Rotate90": Rotate 90 degrees. "Rotate180": Rotate 180 degrees. "Rotate270": Rotate 270 degrees. "RotateAuto": Calculate best rotation automatically.
offsetX	Double	Horizontal offset of the job in mm.
offsetY	Double	Vertical offset of the job in mm.
alignX	String	Horizontal alignment. The following values are supported: "Left" "Center" "Right" Note: If this parameter is specified, the attribute 'offsetX' is ignored.

alignY	String	<p>Vertical alignment.</p> <p>The following values are supported: “Top” “Center” “Bottom”</p> <p>Note: If this parameter is specified, the attribute ‘offsetY’ is ignored.</p>
copyCount	Integer32	Multi output count.
distanceX	Double	Horizontal distance of multiple output in mm.
distanceY	Double	Vertical distance of multiple output in mm.
freeInfo	String	Informational text.
scaleToFit	String	<p>The following values are supported: “FitToPage”: scale to fit a copy per page. “ReduceToFit”: scale-down to fit a copy on a page, don’t scale-up. “FitAllToPage”: scale to fit all copies on a single page. “ClipToMaxPage”: scale to fit a copy per page, crop the image if necessary (aspect ratio of image and media are different).</p> <p>Note: If this parameter is specified, the attributes ‘width’, ‘height’ ‘scaleX’ and ‘scaleY’ are ignored.</p>
mirror	Bool	Flips the output horizontally.
disableAutoTiling	Bool	Controls auto-tiling. If this option is set, no auto-tiling is performed, even if the job size exceeds the media size.
cutmarks	Bool	<p>Controls printing of cut marks.</p> <p>Note: The settings of the cut marks (e.g. type of marks, size and distance) must be specified in the hotfolder.</p>
crop	Bool	Activates/Deactivates cropping.
cropOffsetX cropOffsetY cropWidth cropHeight	Double	If cropping is activated, these parameters are mandatory to define the cropping rectangle in mm.
destinationFolder	String	Sets the output folder for the print files if the printer driver is file based and the naming rule for the output files uses the element “Destination Folder”.
controlWedge	Bool	Activates the output of a control wedge target.
controlWedgeOptions	Object	Options to configure the control wedge. Mandatory if ‘controlWedge’ is true.

Object controlWedgeOptions		
targetName	String	Name of the control wedge target. A list of available targets can be obtained by endpoint 10.1.
fileName	String	Filename of the target job. This corresponds with 'targetName'.
positionLeft positionRight positionTop positionBottom positionAuto turnWithJob mirror turnTop2MediaEdge	Bool	Optional: Positioning options for the target.
align	String	Optional: Alignment of the target. Available options: "Left", "Center", "Right"

4.5.1.4 Printer settings

The settings described in this section are related to the Printer tab of the hotfolder properties dialog.

Object printer		
mediaSizeName	String	Specifies the media size as name. The name is converted to media width and size using the media sizes provided by the printer driver.
mediaWidth	Double	Width of the media, used to produce the job, in mm.
mediaHeight	Double	Height of the media, used to produce the job, in mm. Note: If this parameter is set equal/below 0 (zero), the rollfeed option is activated.
mediaCompensationX mediaCompensationY	Double	Factors to compensate variations of the output size, e.g. due to inaccuracies of the output device (1.0 = 100%).
borderless	Bool	Controls borderless printing. Note: This property is effective only if the printer driver supports borderless printing.

4.5.1.5 RIP settings

The settings described in this section are related to the Rip tab of the hotfolder properties dialog.

Object rip		
antiAliasing	String	The following values are supported: "Off" "Text" "TextVector" "GlobalLow" "GlobalHigh"
useDocumentTransparency	Bool	
removeBackground	String	The following values are supported: "Off" "White" "Black"
removeBackgroundRange	Integer32	Only needed if option removeBackground is not set to "Off". Valid values between 0 and 100.
adjustOpacity	Integer32	Valid values between -50 and 50.

4.5.1.6 Printer driver specific settings

This section contains settings that are specific to a particular printer driver. Each setting is a JSON value of type String, Double, Integer32 or Boolean.

A list of supported settings can be found in a TechNote for the specific printer driver.

4.5.1.7 Workflow settings

The settings described in this section are related to the Workflow tab of the hotfolder properties dialog.

Object workflow		
previewMode	String	Enables creation of the preview. The following values are supported: “None”: No preview is calculated. “Simple”: The preview is calculated without applying color management. “Softproof”: Color management is applied while calculating the preview.
previewResolution	Integer32	Maximum size of the preview in pixel.
ripAndPrint	Bool	Rip the job while printing.
autoRip	String	Automatically rip the job after creation. The following values are supported: “False”: Job is stored in the archive without ripping. “True”: Job is ripped after creation. “Pause”: Job is stored in the rip queue and set to pause.
autoPrint	String	Automatically print the job after ripping. The following values are supported: “False”: Job is stored in the archive without printing. “True”: Job is printed after ripping. “Pause”: Job is stored in the print queue and set to pause.
afterPrint	String	Handling of the job after printing. The following values are supported: “SaveJob”: Save the job in the archive. “DeleteJob”: Delete the job. “Delete print data”: Delete print data and save the job in the archive.
costCalc	String	Enables cost calculation for the job. The following values are supported: “False”: Cost calculation is disabled for to the job. “True”: Cost calculation is enabled for the job. “Auto”: Cost calculation is performed after ripping.

4.5.1.8 Job admin data

The following elements are used to set the job administrative values on the Job Data tab.

Object admin		
jobID	String	
customerID	String	
customerName	String	
comment comment2	String	
dueDate	DateTime	

4.5.1.9 Form data

The elements described in this section, are used to replace the text of form fields in PDF files. The list of the form fields is separated from their contents. This reduces redundancies in cases where multiple sets of data are applied to the same PDF file.

This functionality requires a license of the Variable Data Printing Module (VDPM).

Object formData		
fields	Array	A string array with the field names.
sets	Array	An array of string arrays with the form data. The VDPM is limited to single page PDF files and to a single set of replacement values. So only the first set of fields will be evaluated.

Example:

```
"formData": {
  "fields": [
    "name",
    "email"
  ],
  "sets": [
    [ "Sebastian Wegner", "sebastian.wegner@colorgate.com" ],
    [ "Jens Kopp", "jens.kopp@colorgate.com" ]
  ]
}
```

The replacement values are applied to the form fields in the same order as they are specified in the “fields” array. So, in the given sample, the text for field “name” is “Sebastian Wegner”, the text for the field “email” is “sebastian.wegner@colorgate.com”.

4.5.2 Color table

Modify the color table of a job. Only spot colors can be modified. New entries cannot be created and existing entries cannot be deleted.

PUT jobs/jobID/colorTable		
Result:		
See chapter 8.2		

4.6 Job actions

Runs a task depending on the parameter “action”.

PUT jobs/jobID		
Parameters:		
action	String	<p>“rip” = Rips the job. “print” = Prints the job. “delPrint” = Deletes the print data. “ripPreview” = Regenerates preview data. “delPreview” = Deletes preview data. “abort” = Abort ripping or printing</p>
insertOnTop	Bool	Optional: When action = “rip” or “print” a value of true inserts the job on top of the rip or print queue, so it will be ripped/printed with priority.
downloadOutputFiles	Bool	<p>Optional: When action = “print” a value of true makes the output files for the printer available via REST. This is only useful for file-based printers.</p> <p>The files can be downloaded with the “Files” endpoint. You can obtain the file IDs via job status endpoint 4.4.1 or the notification Job.PrintPageFinished.</p>
Result:		
n/a		

5 Endpoint „files“

This endpoint maintains file uploads. Uploaded files will be identified by an ID which will be generated by the server on upload. This ID will be used when creating a new job. This approach eliminates the need for multipart-messages when creating new print jobs (file to print / parameters).

Uploaded files will be removed automatically when the job is created. If no job is created within 10 minutes after upload the file is also removed automatically.

For convenience it is also possible to download or delete a file.

5.1 Upload file

Upload a file. Put the unencoded binary data into the message body (no multipart message) and provide the content-length field to announce the file size.

POST files?filename=MyFile.tiff		
Parameters:		
filename	String	The original filename without path of the uploaded file. Because the message body contains the binary data of the uploaded file this parameter can only be passed as query in the URI path.
Result:		
fileID	Integer32	The ID of the file. Use this when creating a new print job.
filenameOriginal	String	The name of the file as provided in the request.
filenameInternal	String	The name of the file when saved on the server. If the original filename is unique among the uploaded files it is identical to the original filename. Otherwise it will get an additional serial number.

5.2 Download file

Download a file.

GET files/fileID		
Result:		
Binary data of downloaded file.		

5.3 Delete file

Delete a file.

DELETE files/fileID		
Result:		
n/a		

5.4 List files.

Get list of uploaded files. The list only contains files uploaded by the same client.

GET files		
Result:		
files	Array	The list of files.

Array files		
fileID	Integer32	The ID of the file. Use this when creating a new print job.
filenameOriginal	String	The name of the file as provided in the request.
filenameInternal	String	The name of the file when saved on the server. If the original filename is unique among the uploaded files it is identical to the original filename. Otherwise it will get an additional serial number.
clientAddress	String	The address of the uploading client.
uploadTime	String	Time when the file was uploaded.

5.5 Get file information

Get information about the file.

GET files/fileID/info		
Result:		
fileID	Integer32	The ID of the file. Use this when creating a new print job.
filenameOriginal	String	The name of the file as provided in the request.
filenameInternal	String	The name of the file when saved on the server. If the original filename is unique among the uploaded files it is identical to the original filename. Otherwise it will get an additional serial number.
clientAddress	String	The address of the uploading client.
uploadTime	String	Time when the file was uploaded.

6 Endpoint „notificationSubscriptions“

There are two ways to determine if the status of a print job has changed: Polling and push notifications.

Polling means the client is periodically checking the status of a print job by sending a 4.4.1 request and receiving the current job status. This approach is easy to implement, but results in much network traffic.

Notifications are used to avoid such polling tasks. Instead of constantly asking for status changes the client just waits for a notification. This way network traffic only occurs in case of state changes of jobs or queues.

To receive such notifications, the client itself must provide a http service to which the server sends the notifications. To announce this service the client creates a subscription for notifications.

6.1 List subscriptions

Creates a list of all existing subscriptions.

GET notificationSubscriptions		
Result:		
subscriptions	Array	The list of subscriptions.

Array subscriptions		
server	String	The address of the server.
path	String	The endpoint path of the server.
port	Integer32	The port.
secure	Bool	True for https connection, false for http connection.
userName	String	User who created the subscription.
queueName	String	Optional: Only notifications regarding this queue will be sent.

6.2 Create subscription

Creates a new subscription. A client can create more than one subscription, so many different services can be informed about the state change of one print job.

POST notificationSubscriptions		
Parameters:		
server	String	The address of the server.
path	String	Optional: The endpoint path of the server.
port	Integer32	The port to connect to.
secure	Bool	If this parameter is true a secure connection over https is initiated, otherwise an unsecure http-connection will be established.
queueName	String	Optional: Only notifications regarding this queue shall be sent.
authUsername	String	Optional: Username for Basic Authentication.
authPassword	String	Optional: Password for Basic Authentication.
Result:		
n/a		

6.3 Remove subscription

Remove subscription. A single subscription is identified by the authenticated user, the server, the path and the queue name. If you do not provide a parameter all subscriptions matching the other parameters will be removed. If you do not provide any parameter all subscriptions of the user will be removed.

DELETE notificationSubscriptions		
Parameters:		
server	String	Optional: The address of the server.
path	String	Optional: The endpoint path of the server.
queueName	String	Optional: Name of the queue
Result:		
n/a		

6.4 Sent notifications

The notifications sent to the http client are POST requests with a JSON body.

POST		
Body:		
notification	String	The notification identifier. See the following table for possible values.
date	String	Date of the notification in the format YYYY-MM-DD.
time	String	Time of the notification in the format HH:MM:SS.
jobID	String	Optional: The identifier of the job. This will only be sent for the notification identifiers "Job.XXX".
queueName	String	Optional: The name of the queue. This will only be sent for the notification identifiers "Queue.XXX".
data	Array	Optional: A list of textual data. It is present in the notifications Job.ContainerAdd, Job.ContainerRemove and Job.PrintPageFinished, Job.RipGeneralFailure and Job.PrintGeneralFailure.

Array data		
key	String	Identifier of the data value. Possible values are: When notification is Job.ContainerAdd or Job.ContainerRemove: "ContainerName": Name of the container job. When notification is Job.PrintPageFinished: "PageCount": Number of total pages to print. "PageNumber": Number of the current printed page. Specific printer drivers may add further data. See the driver documentation. When notification is Job.RipGeneralFailure or Job.PrintGeneralFailure: "ErrorMsg": Description of the error.
value	String Integer32	Value.

The following notification identifiers exist:

Value	Description
Job.Created	Generated when the job is successfully created by the RIP.
Job.Deleted	Job has been deleted from the RIP, either manually by the user, or automatically by the RIP after successful production w/o the archive option being activated.
Job.RipStarted	Generated before the calculation of the print data is started.
Job.RipFinished	Generated when the print data has been calculated.
Job.RipGeneralFailure	Generated when an unexpected error occurred during the rip process.
Job.RipPrintdataFailure	Generated when an error occurred when ripping the print data.
Job.RipPreviewFailure	Generated when an error occurred when ripping the preview.
Job.RipPrintdataAbort	Generated when ripping the print data was aborted.
Job.RipPreviewAbort	Generated when ripping the preview was aborted.
Job.RipAnalysisFailure	Generated when an error occurred while analyzing the image file.
Job.PrintStarted	Generated before the data transmission to the printer is started.
Job.PrintGeneralError	Generated when an unexpected error occurred during the print process.
Job.PrintPageStarted	Generated before a new page is printed.
Job.PrintPageFinished	Generated when printing of a page is finished.
Job.PrintFinished	Job has been printed. Note: Most printing devices don't provide a feedback when the job is actually printed. For these devices the notification is generated as soon as all data is transmitted to the printer.
Job.CalcPreviewStarted	Generated before the calculation of the job preview is started.
Job.CalcPreviewFinished	Generated when the preview for the job has been calculated.
Job.DocFileMissing	When the document file is not available for the RIP within the timeout period, this notification is generated.
Job.SettingsChanged	When the document file is not available for the RIP within the timeout period, this notification is generated.
Job.CostDataCalculated	This notification is generated either when the job has been printed, or when cost has been estimated before printing. Note: Only available with printer drivers that support ink counting.
Job.CostDataReported	This notification is generated when the printer has completely finished printing the job. Data is provided in two sub-elements (see following tables for details). Note: Only available for printer drivers that support reporting of usage data.

Job.ContainerAdd	This notification is generated when the job has been added to a container. The “comment” field contains the name of the container.
Job.ContainerRemove	This notification is generated when the job has been removed from a container. The “comment” field contains the name of the container.
Queue.Opened	A queue was opened.
Queue.Closed	A queue was closed.
App.Launched	The application was launched.
App.Closed	The application was closed.

The notifications CostDataCalculated and CostDataReported are only available with the optional Cost Calculation Module (CCM) activated and properly configured.

Sample:

```
{
  "date": "2018-10-18",
  "time": "14:37:56",
  "notification": "Job.Created",
  "jobID": "cea14af8-3f87-4257-bbbf-22c10661ff5a62"
}
```

7 Endpoint „profiles“

This endpoint allows controlling the profiling process from an external application.

Basically, a linearization and profiling task is saved as a CCX file in the Productionserver folder. If you access it with this API the CCX file will be opened and can be modified by the endpoints.

It is not possible to keep more than one CCX file open. If you open another CCX file, the current CCX file will be saved and closed. Because of that it is not recommended to switch between different CCX files frequently.

Every endpoint contains the filename of the CCX file as part of the URI. The file extension CCX can be omitted.

Furthermore, the task of linearization and profiling is divided into steps which most endpoints refer to. The name of the step is also appended to the URI. The following steps exist:

- “preCalibration” only needed when sending/retrieving curves.
- “linearization” for the basic linearization.
- “inkSplitting” for ink splitting calculation
- “linearizationOpt” for the additional linearization of the CMYK channels. Only needed when the color mode of the MIM contains transfer channels.
- “inkLimit” for ink limit determination.
- “profiling” for the profiling task.

7.1 Create profile

Creates a CCX file and the associated MIM.

POST profiles/fileName		
Parameters:		
queueName	String	Name of the queue for which the new profile is created.
mim	Object	Name of the MIM to be created.
hotfolder	String	Name of the hotfolder whose color management settings are used to initialize new MIM. As an alternative, parameter "baseMim" can be used.
baseMim	Object	MIM that is used a template for the new MIM. As an alternative, parameter "hotfolder" can be used.
measurementDevice	String	Name of the device used to measure the targets.
Result:		
n/a		

The mim setting must always specify all three fields.

Object mim		
media	String	Media name of requested MIM-combination.
ink	String	Ink name of requested MIM-combination.
metaMode	String	Metamode name of requested MIM-combination.

7.2 Close

Finishes a linearization task and closes the file.

PUT profiles/filename/close		
Parameters:		
n/a		

7.3 Get status

Gets the status of a linearization task. This endpoint can be used to check repeatedly if a printing or a calculation task has finished.

GET profiles/fileName/status		
Result:		
state	String	Possible values are: "Ready" - Ready to receive new messages. "Printing" - Currently printing a target. "Calculating" - Currently calculating a linearization or profile.
calculationError	String	Optional: The error message of the last calculation task. Only present if the last calculation task was unsuccessful.
percent	Integer32	Optional: a value between 0 and 100 to indicate the progress of the task.
taskStatus	Array	A list of tasks with their status. Only filled if "state" is "Ready".
measurementStatus	String	Optional: If an embedded measurement device is used, this is the status of the measurement. It will only be reported if state is "Ready". Possible values are: "Measuring" - Measuring is in progress. This status will be very rare, because most of the measuring time the overall state is "Printing" even if the printer is already measuring. "Failed" - Measuring failed. "Success" - Measuring succeeded.
measurementFileID	Integer32	Optional: If measuring succeeded the resulting CGATs file is uploaded to the files endpoint and can then be used in endpoint 7.9 to send it to the profile.

Array status		
task	String	<p>Name of the task. Possible values are:</p> <p>“PreCalibration”: Precalibration curves were sent.</p> <p>“DropletSeparation”: Droplet separation curves were sent.</p> <p>“LinearizationTargetCreated”: A linearization target was created.</p> <p>“LinearizationTargetMeasured”: Measurement data for linearization was sent.</p> <p>“LinearizationCalculated”: Linearization curves were calculated.</p> <p>“InkSplitting”: Ink splitting options are set. Only needed when the color mode contains transfer channels.</p> <p>“LinearizationOptTargetCreated”: The additional linearization target of the CMYK channels was created. Only needed when the color mode contains transfer channels.</p> <p>“LinearizationOptTargetMeasured”: Measurement data for additional linearization was sent.</p> <p>“LinearizationOptCalculated”: Additional linearization curves were calculated.</p> <p>“InkLimitTargetCreated”: An ink limit target was created.</p> <p>“ProfileTargetCreated”: Profiling target was created.</p> <p>“ProfileTargetMeasured”: Profiling target was measured.</p> <p>“ProfileCreated”: Profile was created.</p>
status	Bool	Status of task. True, if the task is done / if data exists.

7.4 Get curves

There are some curves involved in the linearization task. Some of them can be configured by the user and therefore can be sent to the Productionserver. All curves are then calculated, and their single points can be retrieved to display the curve to the user. This endpoint gets a list of curves for each channel.

“curveType” must be one of the following:

- “preCalibration” for the calculated precalibration curves.
- “preCalibrationSetup” for the configured precalibration curves.
- “dropletSeparation” for the calculated droplet separation curves.
- “dropletSeparationSetup” for the configured droplet separation curves.
- “linearization” for the basic linearization.
- “linearizationOpt” for the additional linearization of the CMYK channels. Only needed when the color mode of the MIM contains transfer channels.

GET fileName/curves/curveType		
Result:		
curves	Array	List of curves for each channel.

Array curves		
channelName	String	Name of the channel which the curve is applied to. If it is empty the curve is applied to all channels. Possible values are: “Cyan”, “Magenta”, “Yellow”, “Black”, “CyanLight”, “MagentaLight”, “YellowLight”, “BlackLight”, “Red”, “Green”, “Blue”, “Violet”, “Orange” and more.
dotSize	Integer	Only if curveType is “dropletSeparation”: The dot size from 1 (big) to 7 (small). The maximum value depends on the printer configuration.
curvePoints	Array	Array of curve points.

Array curvePoints		
type	String	Type of the point. “Bezier” for a supporting Bezier point smoothing the curve. “Corner” for a point defining the curve.
x	Double	X coordinate of the point between 0 and 1.
y	Double	Y coordinate of the point between 0 and 1.

7.5 Send curves

Sends a list of curves used for the linearization. Sending of pre-calibration curves and droplet separation curves is supported, so “curveType” must be “preCalibrationSetup” or “dropletSeparationSetup”.

Sending curves always resets all existing curves, so if the array is empty all existing curves will be removed.

PUT fileName/curves/curveType		
curves	Array	List of curves.

Array curves		
channelName	String	Name of the channel which the curve is for. If it is empty the values apply to all channels. See chapter 0 for possible channel names.
dotSize	Integer	Only if curveType is “dropletSeparation”: The dot size from 1 (big) to 7 (small). The maximum value depends on the printer configuration.
curvePoints	Array	Array of at least two curve points.

Object curvePoints		
type	String	Type of the point: “Bezier” for a Bezier point smoothing the curve. “Corner” for a supporting point defining the curve.
x	Double	X coordinate of the point between 0 and 1.
y	Double	Y coordinate of the point between 0 and 1.

7.6 List targets

Gets a list of all targets usable with the combination of measurement device and color mode.

GET profiles/fileName/targets/step		
Result:		
targets	Array	Array of all targets.

Array targets		
Result:		
name	String	Name of the target. This is the key to refer to in other endpoints.
contentType	String	Depending on the content type there are different parameters to provide when printing out the target. See chapter 7.7. The following types exist: “Static”: The target is fully described by the ctx file. It is not possible to configure any parameters. “DynFiles”: The layout parameters of the target can be defined by the user. The number of patches and the patch colors are defined in the ctx file. The final target is generated by the Productionserver dynamically. Only available for linearization targets. “DynPatches”: The layout parameters of the target and the number of patches can be defined by the user. The patch colors and the final target is generated by the Productionserver dynamically. Only available for profiling targets.
description	String	Localized target description which can be displayed to the user. The preferred language of the description can be defined by the HTTP header Accept-Language.

7.7 Create target

Creates a target. This must be done before printing a target or sending a pre-calibration curve.

PUT profiles/filename/createTarget/step		
Parameters:		
targetName	String	Name of the target to be used. It must be one of the names listed by endpoint 7.6.
targetLayout	Object	Optional: Specifies the layout parameters to generate the target dynamically. Only appropriate if the type of the target is “DynFiles” or “DynPatches”.
targetOptions	Object	Optional: Specifies some target options to generate the target dynamically. Always needed if the type of the target is “DynPatches”.

Object targetLayout		
targetWidth	Double	Width of the target in mm.
targetHeight	Double	Height of the target in mm.
patchWidth	Double	Width of a patch in mm.
patchHeight	Double	Height of a patch in mm.
rowDistance	Double	Distance of the rows in mm.
patchDistance	Double	Distance of the patches in a row in mm.
columnNames	Bool	Activates printing of column names.
rowNames	Bool	Activates printing of row names.
redundantPatches	Bool	Optional for contentType “DynPatches”: Activates creation of redundant patches to detect inhomogeneities of the substrate.
randomizePatches	Bool	Optional for contentType “DynPatches”: Activates randomizing of patch colors.
bwSeparatorBars	Bool	Optional for certain measurement devices: Activates printing of black/white patch separators.
pages	Array	Read-only: Only when retrieving target data (Chapter 7.10). Contains information about the pages of the target. Useful when dealing with multiple targets.

Array pages		
columns	Integer32	Number of columns.
rows	Integer32	Number of rows.
firstPatchIndex	Integer32	Index of the first patch in the total number of patches.
countPatches	Integer32	Number of patches (Can be less than columns * rows on the last page).

Object targetOptions		
patchCount	Integer32	Optional: Number of patches to print. Only needed if “pageCount” and “patchFile” is empty.
adjustPatchCount	Bool	Optional: Adjust patch count automatically to fill rows and columns.
pageCount	Integer32	Optional: Print these numbers of pages. Only needed if “patchCount” and “patchFile” is empty.
patchFileID	Integer32	Optional: ID of a IT8 file containing the reference patches. You get this number when uploading the file to the “files” endpoint. Only needed if “patchCount” and “pageCount” is empty.
fullGamut	Bool	Optional: Support full gamut of output device.
patchInkLimit	Integer32	Optional: Ink limit for the patches between 100 and 400.

7.8 Print target

Starts the print of a target. This endpoint immediately returns. Send requests to the “status” endpoint to check if the print has finished.

If an embedded measurement device (such as the Epson SpectroProofer) is used, the target will be measured after printing.

PUT profiles/filename/startPrintTarget/step		
Parameters:		
copyCount	Integer32	Optional: Print multiple copies. Default is 1.
copyDistX	Double	Optional: Only used if copyCount is bigger than 1. Horizontal distance of copies in mm, Default is 10.
copyDistY	Double	Optional: Only used if copyCount is bigger than 1. Vertical distance of copies in mm, Default is 10.
rotation	String	Optional. The following values are supported: “Rotate0”: Do not rotate. “Rotate90”: Rotate 90 degrees. “Rotate180”: Rotate 180 degrees. “Rotate270”: Rotate 270 degrees. “RotateAuto”: Calculate best rotation automatically. Default is “Rotate0”.
printJobInfo	Bool	Optional: Print job info. Default is false.
centerX	Bool	Optional: Center target horizontally. Default is false.
mirror	Bool	Optional: Mirror output. Default is false.
jobName	String	Optional: The name of the job. On file-based printer drivers this may also affect the output file names. If this parameter is missing a standard job name will be generated automatically.
getTargetFiles	Bool	Optional: The target PDFs shall be provided via “files” endpoint. The result will contain the file IDs then. Default is false.
Result:		
jobID	String	The identifier of the job. Use this in subsequent requests regarding this job.
jobName	String	The name of the job.
fileName	String	The name of the file to print.
size	String	The size of the job.
copies	Integer32	The number of copies.
fileSize	String	File size of the job.
targetFiles	Array	Only if “getTargetFiles” is true. Provides the IDs and names of the target file PDFs. They can then be downloaded via “files” endpoint.

7.9 Send data

Send data.

PUT profiles/filename/data/step		
Parameters:		
fileID	Integer32	Optional: ID of the associated IT8 file. You get this number when uploading the file to the “files” endpoint. Not needed for InkLimit or when sending patchData.
patchData	Array	Optional: Patch data.
inkLimitOptions	Object	Optional: Options for the ink limit.
inkSplittingOptions	Object	Optional: Options for the ink splitting. Only needed when the color mode of the MIM contains transfer channels.

Array patchData		
channels	Array	Optional: only if the target contains more than one channel (step linearisation).
patches	Array	Optional: only if the target contains only one channel (step profiling).

Array channels		
channelName	String	Name of the channel. See chapter 0 for possible channel names.
maxDensity	Double	Optional: The full color value of the channel.
maxDensityIndex	Integer32	Optional: The patch index of the full color value. Alternative to “maxDensity”.
patches	Array	Patch values.

Array patches		
index	Integer32	Index of the patch. It is sufficient to provide only the patches you want to change instead of providing all patches of the channel.
ignoreSample	Bool	True if the patch measurement of this patch should be ignored when calculating a linearization or profile.

Object inkLimitOptions		
inkLimitType	String	Possible values: “AllChannels” “KeepBlack”
maxDensity	Integer32	Value of the maximum density between 0 and 400 (for CMYK).

Object inkSplittingOptions		
channels	String	Optional: Array of inkSplittingOptions. Only needed if options should be applied per channel. In this case all other options can be omitted here.
preset	String	Optional: Name of the preset. If no preset is specified, you must specify the following values. The following presets exist: “Minimum Ink (Default)” “Maximum Density” “Enhanced Smoothness” “Balanced”
channelName	String	Optional: Name of the channel (eg. “Cyan”, “Magenta”). Only used if options are sent for each channel in “channels”,
lightLimit	Integer32	Value of the maximum light ink amount between 0 and 100.
totalLimit	Integer32	Value of the maximum ink amount between 0 and 100.
transition	Integer32	
cycles	Integer32	
smoothness	Integer32	
dMax	Integer32	

7.10 Get data

Get reference and measurement data. The data itself is stored in the fileID endpoint and can be downloaded via endpoint 5.2.

GET profiles/filename/data/step		
Result:		
fileIDReference	Integer32	ID of the IT8 file containing the reference values.
fileIDMeasurement	Integer32	Optional: Only if measurement exists. ID of the IT8 file containing the measurement values.
targetLayout	Object	Target layout options. See chapter 7.7
profilingOptions	Object	Appears only if step is profiling: Returns the options for the profiling which can be set when calculating a profile. See chapter 7.11.
patchData	Object	Contains the patch data for the target.

Object patchData		
densityMode	String	Mode of the density values. “ISO_A”, “ISO_E” or “ISO_T”.
channels	Array	Appears only if the target contains more than one channel (step linearisation).
patches	Array	Appears only if the target contains only one channel (step profiling).

This array only appears if the target contains more than one channel (step Linearization).

Array channels		
channelName	String	Name of the channel. See chapter 0 for possible channel names.
maxDensity	Double	The full color value of the channel.
maxDensityIndex	Integer32	The patch index of the full color value.
patches	Array	Patch values.

Array patches		
index	Integer32	Index of the patch.
name	String	Name of the patch.
measured	Bool	True if the patch contains measurement data.
ignoreSample	Bool	True if the patch measurement of this patch should be ignored when calculating a linearization.
lab	Array<Double>	Only if patch has measurement data: The lab values of the measurement.
lch	Array<Double>	Only if patch has measurement data: The c value of the measurement.
density	Double	Only if patch has measurement data: The density of the patch.
rgb	Array<Double>	Only if patch has measurement data: An RGB color value to display the patch.
spectralValues	Array<Double>	Only if patch has spectral measurement data: The spectral values of the measurement.
refValues	Array<Double>	Only if patch has reference data.
deltaE	Double	Only if patch has reference data and measurement data. The deltaE between reference and measurement.

7.11 Calculate linearization / profile

Starts the calculation of linearization curves or of a profile. This endpoint immediately returns. Send requests to the “status” endpoint to check if the calculation has finished.

PUT profiles/filename/startCalculate/step		
Parameters:		
linearizationOptions	Array	Optional: Options for the linearization calculation.
profilingOptions	Object	Optional: Mandatory options for the profile calculation.

Array linearizationOptions		
channelName	String	Name of the channel the smoothing is applied to. If the name is empty this value applies for all channels which do not have an individual smoothing set. See chapter 0 for possible channel names.
smoothing	Integer32	Optional: Value of the smoothing value between 0 and 100.

Object profilingOptions		
blackGeneration	Object	Optional: Black generation options.
blackPointOptions	Object	Optional: Black point mode.
options	Object	General profiling options.
separationCurves	Array	Curves of the separation preview, read only. Same data type as array “curves”, chapter 7.4.

Object blackGeneration		
multicolorMode	String	Optional, default value = "Smooth". Available options: "Strong" "Smooth" "Sparse" "SmoothSplit"
blackGeneration	String	Available options: "UCR" "GCR" "UCRSmooth" "GCRSmooth" "MaxK"
gcrStrength	Integer32	Value between 0 and 100.
blackStart	Integer32	Value between 0 and 100.
blackWidth	Integer32	Value between 0 and 100.
pureBlack	Bool	Optional, default value = false.
pureGray	Bool	Optional, default value = false.

Object blackPointOptions		
blackPointMode	String	"Auto" "Lock" "BalanceCMY"
blackMax	Integer32	Only needed if blackPointMode is not "Lock"
inkTotal	Integer32	Only needed if blackPointMode is not "Lock"
blackPointValues	Array<Integer32>	Only needed if blackPointMode is "Lock"

Object options		
viewCondition	String	Optional, default value = "D50". Viewing light condition. Possible value: "D50" "D55" "D65" "Tungsten" "IlluminantA", "IlluminantB", "IlluminantC", "IlluminantE" "IlluminantF2", "IlluminantF7", "IlluminantF11", "IlluminantF12"
applyToWindows	Bool	Optional, default value = false Make profile available for Windows color management.
inkTotal	Integer32	Optional, default value = 400.
profileSize	String	Optional, default value = "Large". Possible values: "Small", "Medium", "Large", "Huge"
profileFormat	Integer32	Optional, default value = 2 2 or 4.
gamutMapping	String	Optional, default value = "Neutral". "Neutral", "ColorBoost", "AbsCompression"
grayBalance	Integer32	Optional, default value = 0 Valid values are between -10 and 10.
channelsActive	Array	Optional: List of active channels.
channelsCombinable	Array	Optional for generic color mode: List of active combinable channels.
combineChannelsAuto	Bool	Optional for generic color mode
brightenerCorrection	Bool	Optional, default value = false.

measurementCorrection	Bool	Optional, default value = false.
coloredSubstrateOptimization	Bool	Optional, default value = false.
applyLinToProfile	Bool	Optional, default value = false.

Array channelsActive / channelsCombinable

channelName	String	Name of the channel.
active	Bool	Status of the channel.

7.12 Open profile

Opens a CCX file in a specific queue. Since accessing any profile-endpoint implicitly opens a ccx this endpoint is only needed if you have more than one queue of the same printer type. Without calling this endpoint the ccx would be opened in the first queue of the associated printer.

PUT profiles/filename/open		
Parameters:		
queueName	String	Name of the queue for which the profile shall be opened.
Result:		
n/a		

7.13 Workflow examples

Examples for the consecutive steps of a linearization

CMYK	
7.1	Create profile (Not needed when continuing a previously created profile.)
7.5	Optional: Send precalibration curves
7.7	Create linearization target
7.8	Print linearization target
7.3	Poll status until "Ready"
	Measure linearization target externally
5.1	Upload IT8 measurement file
7.9	Send measurement data of linearization target
0	Calculate linearization curves
7.3	Poll status until "Ready"
7.7	Print ink limit target
7.8	Print ink limit target
7.9	Send ink limit data
7.3	Poll status until "Ready"
7.7	Create profiling target
7.8	Print profiling target
7.3	Poll status until "Ready"
	Measure profiling target externally
5.1	Upload IT8 measurement file

7.9	Send measurement data of profiling target
0	Calculate profile
7.3	Poll status until “Ready”
7.2	Close profile

CMYK + transfer channels	
7.1	Create profile (Not needed when continuing a previously created profile.)
7.5	Optional: Send precalibration curves
7.7	Create linearization target
7.8	Print linearization target
7.3	Poll status until "Ready"
	Measure linearization target externally
5.1	Upload IT8 measurement file
7.9	Send measurement data of linearization target
0	Calculate linearization curves
7.3	Poll status until "Ready"
7.7	Create CMYK linearization target
7.8	Print CMYK linearization target
7.3	Poll status until "Ready"
	Measure CMYK linearization target externally
5.1	Upload IT8 measurement file
7.9	Send measurement data of CMYK linearization target
0	Calculate CMYK linearization curves
7.3	Poll status until "Ready"
7.7	Create profiling target
7.8	Print linearization target
7.3	Poll status until "Ready"
	Measure profiling target externally
5.1	Upload IT8 measurement file
7.9	Send measurement data of profiling target
0	Calculate profile
7.3	Poll status until "Ready"
7.2	Close profile

8 Endpoint „colorTables“

This endpoint accesses color table files found in the “ColorTables” folder of the installation. The structures used to display and modify a color table are the same when requesting or modifying the color table of a job (see 4.4.6 and 4.5.2).

8.1 List color tables

Gets a list of all color table files in the “ColorTables” folder of the installation. The file extension “cct” is omitted.

GET colorTables		
Result:		
colorTables	Array	The list of all color table names.

8.2 Get color table

Gets a list of all color table entries of a color table.

GET colorTables/colorTableName		
Result:		
colorTableEntries	Array	The list of all color table entries.

Values that are invalid for a certain configuration will not be displayed when getting a color table file. When changing a color table those values will not be accepted.

Array colorTableEntries		
name	String	This is the identifier to address a particular entry. For spot colors it is the name of the spot color. All other colors build the name of the input components, eg RGB (255/0/0).
enabled	Bool	True if replacement is enabled
enabledBitmap	Bool	Valid only for color table files and spot colors of rip jobs.
enabledVec	Bool	Valid only for color table files and spot colors of PDF rip jobs.
enabledText	Bool	Valid only for color table files and spot colors of PDF rip jobs.
spotColorMaxGamut	Bool	Valid only for spot colors with output color model “Lab” or “Spectral”.
sourceName	String	Read-Only: Valid only for PantoneLIVE® colors.
masterName	String	Read-Only: Valid only for PantoneLIVE® colors.
inputColor	Object	The input color for this entry.
outputColor	Object	The output color for this entry.

documentColor	Object	The document color for this entry. Appears only when retrieving the color table of a PDF rip job.
---------------	--------	---

Object inputColor

colorModel	String	The color model name of the input color. Allowed values: “Spot”, “CMYK”, “RGB”, “Gray”, “CMYKRange”, “RGBRange”, “GrayRange”
values	Array	Double array of component values. Only applicable if the color model is not “Spot”.

Object outputColor

colorModel	String	The color model name of the output color. Allowed values: “Lab”, “Spectral”, “CMYK”, “Alias”, “None”, “Omit”, “DeviceSpot”, “Custom”, (Device colors) “Alias”
customColorModel	String	The name of the device color model. Only applicable if colorModel is “Custom”.
aliasName	String	The name of the alias color. Only applicable if colorModel is “Alias”.
opacity	Double	Opacity value between 0 and 100. Only applicable for spot colors with output color model “Lab” or “Spectral”.
samples	Array	Sample values. Only applicable for color model “CMYK”, “Lab” or “Spectral”.

Array samples		
tint	Integer32	Optional for color model “Lab” and “Spectral”: Tint value for the sample between 0 and 100. Can be omitted if the color has only one sample. The first sample must always have a tint value of 100.
backing	Integer32	Optional for color model “Lab” and “Spectral”: Backing value for the sample, “Substrate” or “Black”. Can be omitted if the color has only one sample. The first sample must always have backing “Substrate”.
values	Array	Double array of component values.
spectralValues	Array	Optional: Only for color model “Spectral”: Double array of 36 spectral values (380-730nm in 10nm steps).
predictedColor	Object	Optional: Only if an output profile is active.
deltaE76 deltaE94 deltaE00	Double	Optional: Only if an output profile is active. Delta-E distance of predicted color.
replacementSample	Object	Sample values to display the replacement color.
predictedSample	Object	Optional: Only if an output profile is active. Sample values to display the predicted color.

Object predictedColor		
type	String	Type of the color values. Always “LAB”
values	Array	Three Lab values.

Object replacementSample, predictedSample		
type	String	Type of the color values. Always “RGB”
values	Array	Three RGB values.

Object documentColor		
colorModel	String	The color model name of the document color.
values	Array	Double array of component values.
replacementSample	Object	Optional: Only if an output profile is active. Sample values to display the document color.

8.3 Create color table entries

Creates new color table entries. If the color table does not yet exist it is created. The new entries must not already exist in the color table.

POST colorTables/colorTableName		
Parameters:		
colorTableEntries	Array	Array of new entries. See chapter 8.2

8.4 Change color table entries

Changes color table entries in an existing color table. Values which are not provided will be left unchanged.

PUT colorTables/colorTableName		
Parameters:		
colorTableEntries	Array	Array of changed entries. See chapter 8.2

8.5 Delete color table / entries

Deletes color table entries or a color table file.

DELETE colorTables/colorTableName		
Parameters:		
colorTableEntries	Array	Optional: String array of entry names. If no array is provided the complete file will be deleted.

Alternatively, a single entry name can be provided in the URL:

DELETE colorTables/colorTableName/entryName		
Parameters:		
n/a		

9 Endpoint „colorCorrections“

This endpoint allows controlling the functionality of the Color Correction Loop Module (CCLM). Please see section 9.7 for the steps required to do an iterative color correction process.

9.1 Create color correction

Starts the color correction process for a job and defines basic settings.

POST colorCorrections/jobID		
Parameters:		
inspectionSystem	String	Type name of the inspection system used for the color correction. Allowed values: “IPAC ACMS”, “IPAC ICMS”
fileNameSuffix	String	Suffix used for the name of inspection files.

9.2 Delete color correction

Removes the color correction for a job.

DELETE colorCorrections/jobID		
Parameters:		
a/a		

9.3 Get color correction info

Retrieves information about the color correction process for a job.

GET colorCorrections/jobID/info		
Parameters:		
n/a		
Result:		
inspectionSystem	String	Type name of the inspection system used for the color correction. Allowed values: "IPAC ACMS", "IPAC ICMS"
fileNameSuffix	String	Suffix used for the name of inspection files.
iterationCount	Integer32	Number of iterations.
activeIteration	Integer32	1-based index of the active iteration. Set to 0 if there is no active iteration.
iterationFiles	Array	The array contains one element per iteration, with the names of the master and sample data files.
additionalJobs	Array	The list of additional jobs. Only if additional jobs exist.

Array files		
master	String	File name of the master data file.
sample	String	File name of the sample data file.

Array additionalJobs		
jobID	String	The identifier of the job.
jobName	String	The name of the job.
fileName	String	The name of the file.

9.4 Create inspection system files

Starts the creation of the image files that the inspection system requires to acquire color values for the job. Use the endpoint "Get status" (9.7), to check when the task is finished.

POST colorCorrections/jobID/inspectionFiles		
Parameters:		
n/a		

9.5 Create iteration

Adds an iteration to the color correction process for a job and starts the calculation of the correction profile as well as a new set of inspection system files.

The master and sample data files must be uploaded before the iteration is created, see section 5.1 “Upload file”.

Use the endpoint “Get status“, to check when the task is finished (section 9.7).

POST colorCorrections/jobID/iteration		
Parameters:		
masterFileID	Integer32	ID of the master data file.
sampleFileID	Integer32	ID of the sample data file.
printJobWithCorrection	Bool	Optional: If true, the job will be printed after the correction profile was created.
printAdditionalJobsWithCorrection	Array	Optional: The list of additional jobs to print after the correction was applied. These jobs must have been added before with endpoint 9.8.

9.6 Delete iteration

Removes the active iteration from the color correction process for a job.

If there is an earlier iteration, it is activated.

DELETE colorCorrections/jobID/iteration		
Parameters:		
n/a		

9.7 Get status

Gets the status of the running task. This endpoint can be used to check repeatedly if a task has finished.

GET colorCorrections/jobID/status		
Parameters:		
n/a		
Result:		
State	String	Possible values are: “Ready” - The inspection files are ready for download. “CreatingInspectionFiles” - Currently creating the inspection files. “Calculating” - Currently creating correction profile and the inspection files.
percent	Integer32	A value between 0 and 100 to indicate the progress of the task. Only filled on state not “Ready”.
fileID	Integer32	ID of the ZIP with the inspection files. Only filled if file was created and state “Ready”.

9.8 Add jobs to correction

The correction can be applied to additional jobs. This endpoint allows to add jobs to this list. The current correction profile will be applied to all jobs.

POST colorCorrections/jobID/additionalJobs		
Parameters:		
additionalJobs	Array	A string array with jobIDs to be added to the correction.

9.9 Remove jobs from correction

This endpoint allows to remove jobs from the list of additional jobs. The applied correction profile of this job will also be removed.

DELETE colorCorrections/jobID/additionalJobs		
Parameters:		
additionalJobs	Array	A string array with jobIDs to be removed from the correction.

9.10 Workflow example

The following steps are typically used for an iterative color correction process.

Step	Endpoint	
1	9.1	Create color correction
2	9.4	Create inspection system files
3	9.5	Poll status until inspection system files are available
4	5.2	Download ZIP archive with inspection system files
5		External workflow on inspection system to detect color deviations on the printed goods
6	5.1	Upload master and sample data files, provided by the inspection system
7	9.6	Create iteration
8	9.8	Poll status until correction profile and inspection system files are available
9	5.2	Download ZIP archive with inspection system files
		Repeat from step 5 for next iteration

10 Endpoint „controlWedge“

This endpoint allows controlling the functionality of the control wedge evaluation for a job. (CCLM). Please see chapter xxx to see how to configure a job printing a control wedge.

10.1 Get targets

Creates a list of available targets depending on the color mode and the measurement device.

GET controlWedge/jobID/targets		
Parameters:		
measurementDevice	String	Name of the device used to measure the target.
Result:		
targets	Array	The list of available targets.

Array targets		
targetName	String	Name of the target description file.
fileName	String	Filename of the target.
description	String	Descriptive text.

10.2 Get evaluation

Gets the control wedge evaluation for a job.

GET controlWedge/jobID		
Parameters:		
printingCondition	String	Optional for proofing workflow. The printing condition. The list of available conditions are delivered by this endpoint in field 'available-Conditions'.
createReport	Bool	Optional: Creates a html report file. It will be uploaded to the files endpoint. Default is false.
reportUseLogo	Bool	Optional: The html report file will contain a logo. Default is true.
reportDetailed	Bool	Optional: The html report is detailed with patches: Default is true.
Result:		
targetName	String	Name of the target description file.
printingCondition	String	Only in proofing workflow. Printing condition.
printingConditionTitle	String	Only in proofing workflow. Display name for the printing condition.

reference	String	Reference profile.
simulationProfile	String	Simulation profile.
deltaEType	String	DeltaEType. Available values: “DeltaE”, “DeltaE94”, “DeltaE2000”
customer	String	Customer name.
comment	String	Comment.
measured	String	Timestamp of creation.
colorMode	String	Color mode of the job.
results	Array	Results of the evaluation.
reportFile	Object	Contains the data of the html report files if parameter ‘createReport’ is true.
patches	Array	The list of patches.
availableConditions	Array	Proofing workflow: The list of available printing conditions.

Array results

class	String	Name of result class.
values	Array	List of result values.

Array values

name	String	Name of condition.
value	Double	Result value.
limit	Double	Standard limit.
limitOk	Bool	True if ‘result’ is not higher than ‘limit’.
userLimit	Double	User limit.
userLimitOk	Bool	True if ‘result’ is not higher than ‘userLimit’.

Object reportFile

fileIDReport	Integer32	ID of the html report file.
fileNameReport	String	Name of the html report file.
fileIDLogo	Integer32	ID of the logo file.
fileNameLogo	String	Name of the logo file.

Array patches

name	String	Name of the patch.
labRef	Array<Double>	Lab values of the reference.

rgbRef	Array<Double>	An RGB color value to display the reference value.
deviceColor	Array<Double>	Device color values of the reference.
measured	Bool	True if the patch contains measurement data.
ignoreSample	Bool	If measurement data exists: True if the patch measurement of this patch should be ignored when calculating the result.
lab	Array<Double>	Lab values of the measurement.
rgb	Array<Double>	An RGB color value to display the measurement value.
deltaE deltaH deltaAB	Double	Delta between measurement and reference.

Array availableConditions

printingCondition	String	Printing condition.
printingConditionTitle	String	Display name for the printing condition.

10.3 Create evaluation

Creates a new control wedge evaluation for a job. The measurement data file must be uploaded before, see section 5.1 “Upload file”.

POST controlWedge/jobID

Parameters:		
fileID	Integer32	ID of the associated IT8 file. You get this number when uploading the file to the “files” endpoint.
deltaEType	String	Optional: Available values: “DeltaE”, “DeltaE94”, “DeltaE2000”
backing	String	Optional: Available values: “Black”, “White”
customer	String	Optional
comment	String	Optional

10.4 Delete evaluation

Removes the control wedge evaluation from the job.

DELETE controlWedge/jobID		
Parameters:		
n/a		

10.5 Workflow example

The following steps are used to create a control wedge evaluation.

Step	Endpoint	
1	10.1	Get a list of available targets for the job
2	4.5.1.3	Change job settings to configure the control wedge to be printed
3	4.6	Rip and print the job together with the control wedge
4		External workflow to measure the control wedge target
5	5.1	Upload the it8 file with the measurement data
6	10.3	Create the evaluation
7	10.2	Get the evaluation results

11 Endpoint „container“

This endpoint allows creating and manipulating container jobs. It provides only the functionality specific to containers. Since a container is also a job most parts of the job endpoint (See chapter 4) also apply to containers.

11.1 Create container

Creates a container from a list of jobs.

POST container		
Parameters:		
jobs	Array<String>	The list of jobs (job IDs) to be put into the container.
settings	Object	Optional: The settings of the container. See chapter 11.2.2
Result:		
		See contents of Array jobs in chapter 4.1.

11.2 Get container data

11.2.1 Get job list

Gets the list of jobs in the container.

GET container/jobID/jobs		
Parameters:		
n/a		
Result:		
jobs	Array	The list of jobs.

Array jobs		
jobID	String	The identifier of the job.
jobName	String	The name of the job.
fileName	String	Filename of the job.
x	Double	X-offset of the job inside the container.
y	Double	Y-offset of the job inside the container.
width	Double	Job width.
height	Double	Job height.

11.2.2 Get settings

Gets the container settings. Since a container is a special kind of jobs with limited settings this endpoint is also called if you call endpoint 4.4.7 on a container.

GET container/jobID/settings		
Parameters:		
n/a		
Result:		
settings	Object	The settings of the container.

Object settings		
job	Object	Job settings.
workflow	Object	Workflow settings. See Chapter 4.5.1.7.
printer	Object	Printer settings.
container	Object	Container settings.

11.2.2.1 Job settings

The settings described in this section are related to the Printer tab of the hotfolder properties dialog.

Object printer		
jobName	String	The name of the job.

11.2.2.2 Printer settings

The settings described in this section are related to the Printer tab of the hotfolder properties dialog.

Object printer		
mediaSizeName	String	Specifies the media size as name. The name is converted to media width and size using the media sizes provided by the printer driver.
mediaWidth	Double	Width of the media, used to produce the job, in mm.
mediaHeight	Double	Height of the media, used to produce the job, in mm. Note: If this parameter is set equal/below 0 (zero), the rollfeed option is activated.
borderless	Bool	Controls borderless printing. Note: This property is effective only if the printer driver supports borderless printing.

11.2.2.3 Container settings

The settings described in this section are related to the Container tab of the hotfolder properties dialog.

Object container		
arrangeMethod	String	“NONE”: No automatic arrangement “EDGE2EDGE”: Cut optimized “ANYTYPE”: Media optimized “ORDERED”: Ordered nesting “TRIM”: Trim nesting
jobDistanceX jobDistanceY	Double	The distance between jobs when they are automatically arranged.
fixedSize	Bool	Forces the size of the container to be fixed.
width height	Double	Fixed container size in mm. Only applicable when fixedSize=true.
marginTop marginBottom marginLeft marginRight	Double	Margins in mm.
alignX	String	Horizontal alignment. The following values are supported: “Left” “Center” “Right”
copyCount	Integer32	Multi output count.
copyDistX	Double	Horizontal distance of multiple output in mm.
copyDistY	Double	Vertical distance of multiple output in mm.

11.3 Change container data

11.3.1 Add jobs

Adds new jobs to an existing container.

PUT container/jobID/jobs		
Parameters:		
jobs	Array<String>	The list of jobs (job IDs) to be added to the container.
Result:		
n/a		

11.3.2 Change settings

Changes the settings of a container. Since a container is a special kind of jobs with limited settings this endpoint is also called if you call endpoint 4.5.1 on a container.

PUT container/jobID/settings		
Parameters:		
settings	Object	See chapter 11.2.2 for the options to set.
Result:		
n/a		

11.3.3 Change positions

Changes the positions of jobs in a container. Overrides the automatic arrangement of job positions.

PUT container/jobID/settings		
Parameters:		
jobs	Array	Array of jobs.
Result:		
n/a		

Array jobs		
jobID	String	The identifier of the job.
x	Double	X-offset of the job inside the container.
y	Double	Y-offset of the job inside the container.

11.3.4 Trigger auto arrangement

Triggers the automatic arrangement of jobs.

PUT container/jobID/autoArrange		
Parameters:		
n/a		
Result:		
n/a		

11.4 Remove container data

11.4.1 Remove jobs

Removes jobs from a container. The jobs are put back into the job archive.

DELETE container/jobID/jobs		
Parameters:		
jobs	Array<String>	The list of jobs (job IDs) to be removed from the container.
Result:		
n/a		

11.4.2 Split container

Removes all jobs from a container and then removes the empty container. The jobs are put back into the job archive.

DELETE container/jobID		
Parameters:		
n/a		
Result:		
n/a		

12 Testing with Postman

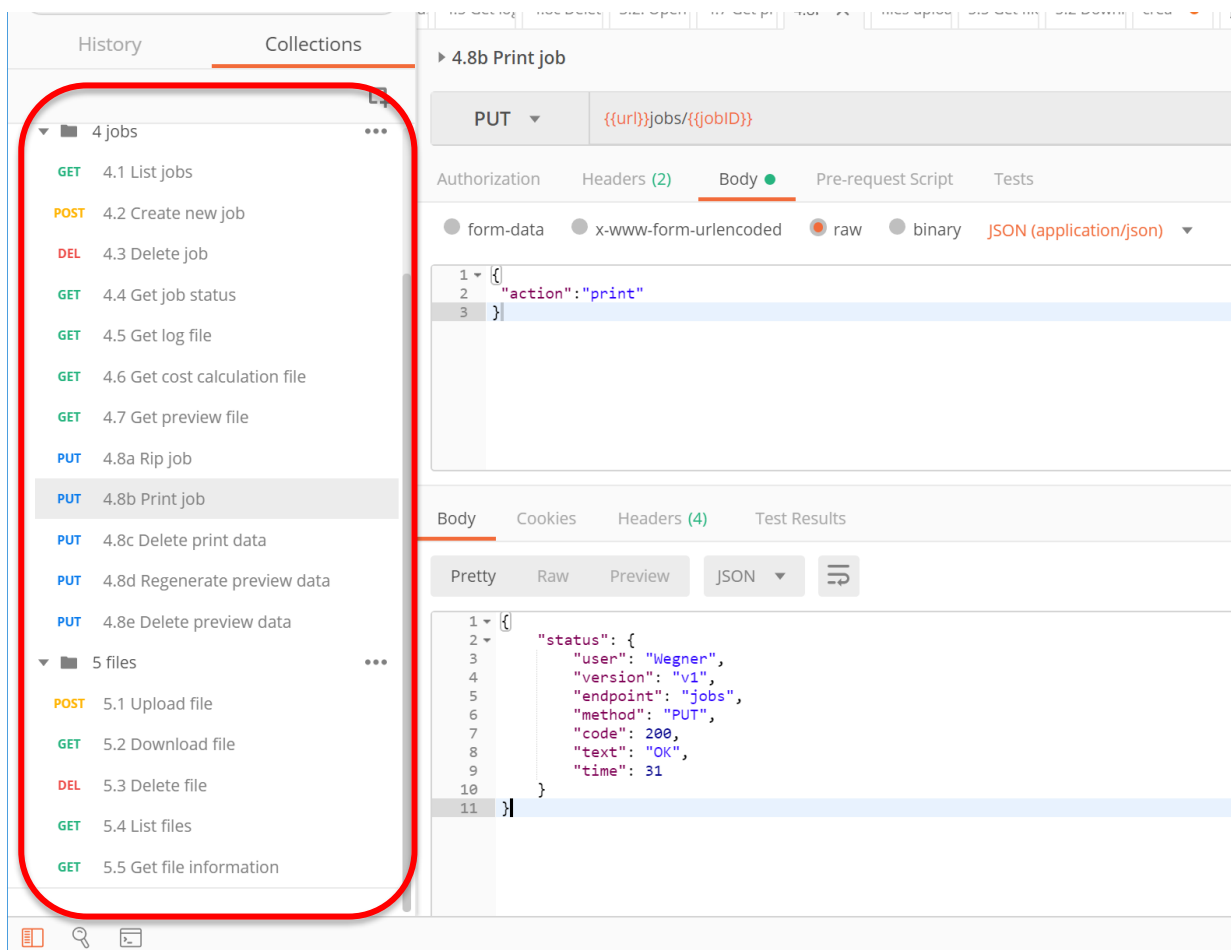
12.1 Introduction

[Postman](#) is a tool to manage HTTP requests and send requests to servers. This way it is possible to see the API in action before implementing your own API client code. This is no manual for Postman but explains some basic steps you must do to test the API with it.

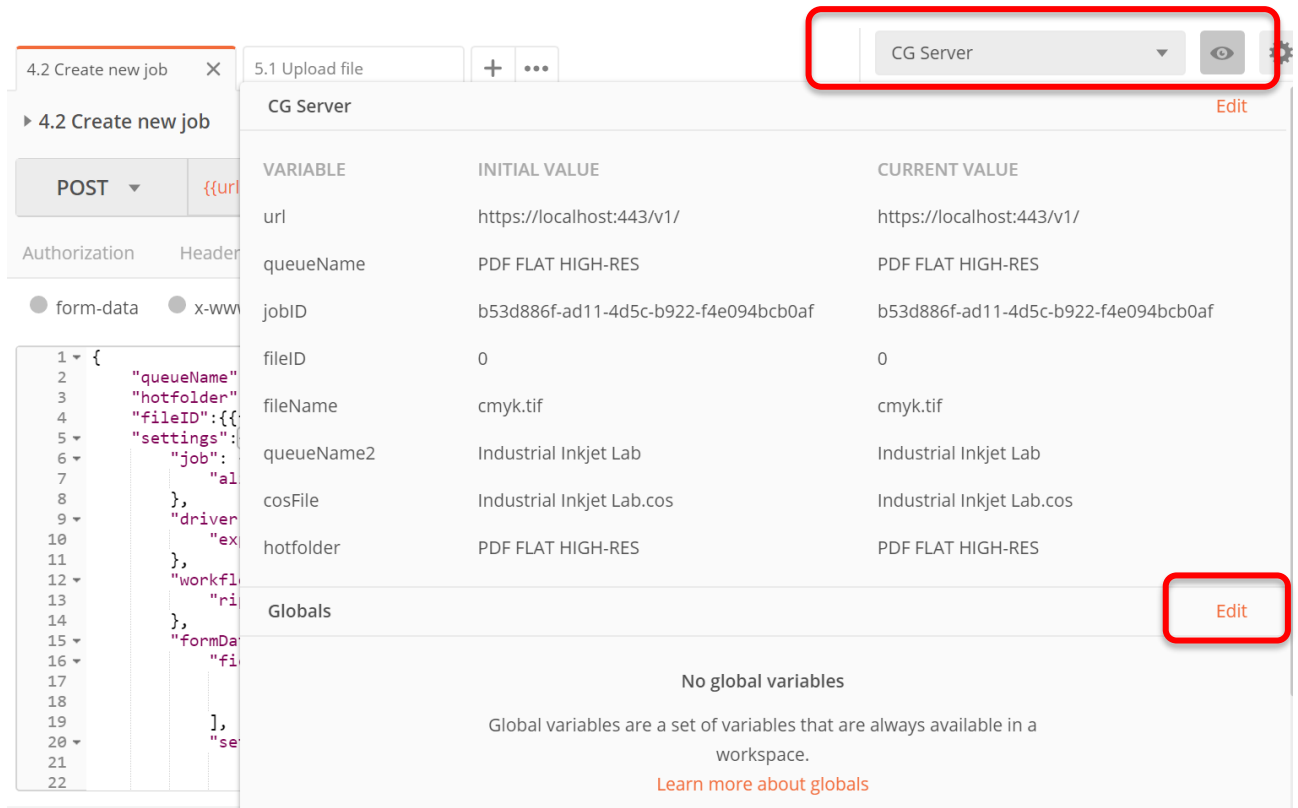
You find a set of requests in the associated json file **CGWebAPI_PostmanCollection.json** which you can import as a collection in Postman. Select it with the menu command File->Import to load it into Postman.

There is also a json file **CGWebAPI_PostmanEnvironment.json** with environment variables you will need. Import it the same way as the collection.

After that you see the requests at the left:



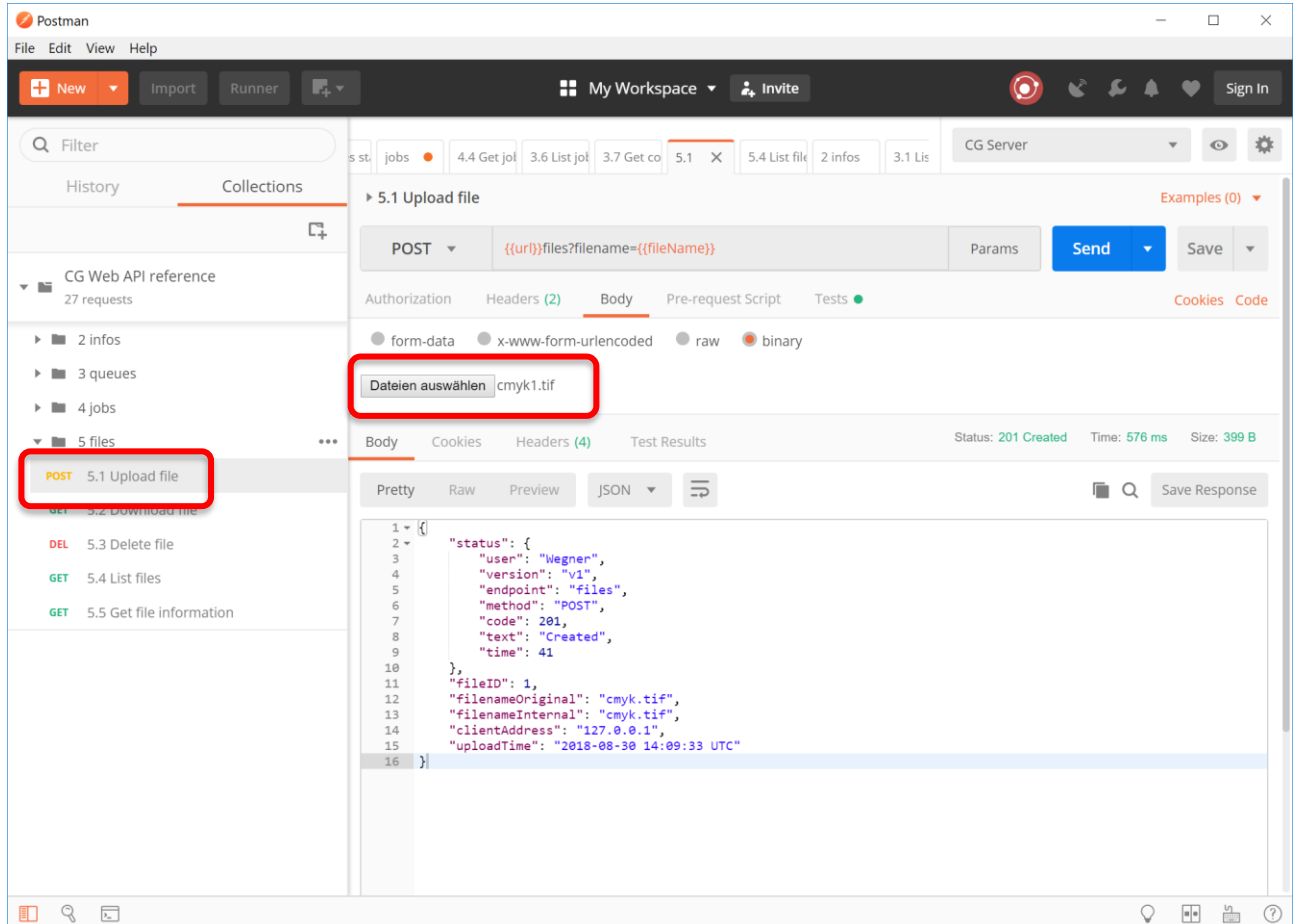
The environment is named “CG Server”. It should be visible at the top of the window. If you click the eye symbol the list of variables in the environment is visible. You may have to edit some of the initial values if your server address differs or if your queues have different names.



The following environment variables are defined:

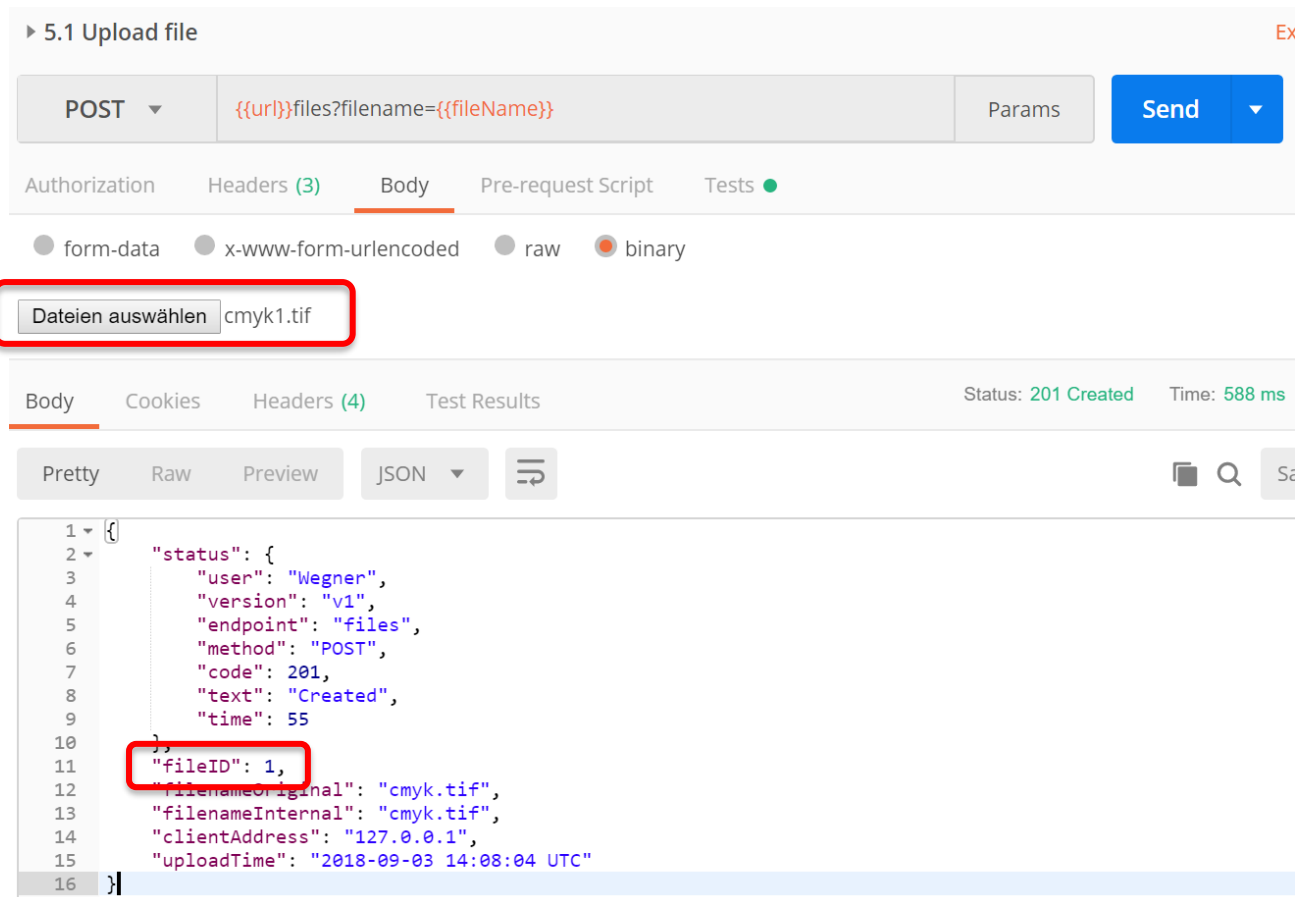
Variable	Description
url	The url to the server including the version segment.
queueName	The name of the queue you want to put new jobs into. It should already be opened in the Productionserver.
jobID	The ID of the job you want to manipulate. Will be set automatically to the new job when created with a 4.2 request.
fileID	The ID of the last uploaded file. Will be set automatically if you upload a file with a 5.1 request.
fileName	Should be the filename you selected in the 5.1 request. Set this parameter here.
queueName2	The name of another queue you want to test opening (3.2) and closing (3.3) queues with.
cosFile	The filename of queueName2. Will be used of request 0 to open another queue.
hotfolder	The name of the hotfolder used when a new job is created. Must be configured in the queue selected by queueName.

To print a file, you have to upload it. Before sending the first file upload request you must select a printable file in the 5.1 request.



12.2 Example: Printing a file:

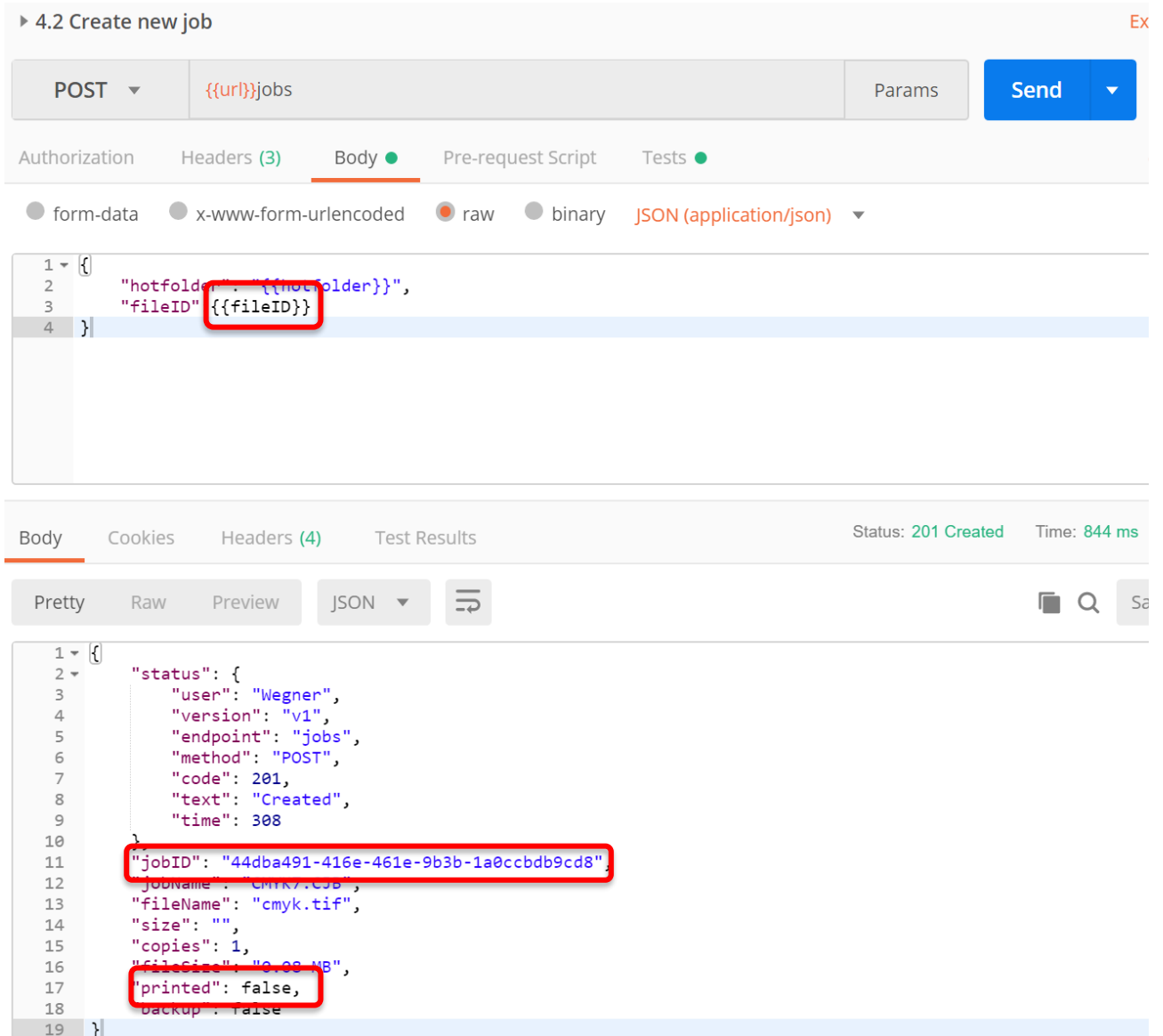
Send a 5.1 request to upload the file. You receive the fileID.



The screenshot displays a REST client interface for a request labeled "5.1 Upload file". The request method is POST, and the URL is `{{url}}files?filename={{fileName}}`. The request body is set to "binary" and a file named "cmyk1.tif" is selected, as indicated by the "Dateien auswählen" button and the filename in the input field. The response status is 201 Created, and the response body is a JSON object:

```
1 {
2   "status": {
3     "user": "Wegner",
4     "version": "v1",
5     "endpoint": "files",
6     "method": "POST",
7     "code": 201,
8     "text": "Created",
9     "time": 55
10  },
11  "fileID": 1,
12  "filenameOriginal": "cmyk.tif",
13  "filenameInternal": "cmyk.tif",
14  "clientAddress": "127.0.0.1",
15  "uploadTime": "2018-09-03 14:08:04 UTC"
16 }
```

Send a 4.2 request to create a new print job. You provide the fileID you received in the previous step. You receive the jobID of the newly created job. The status “printed” is set to false.

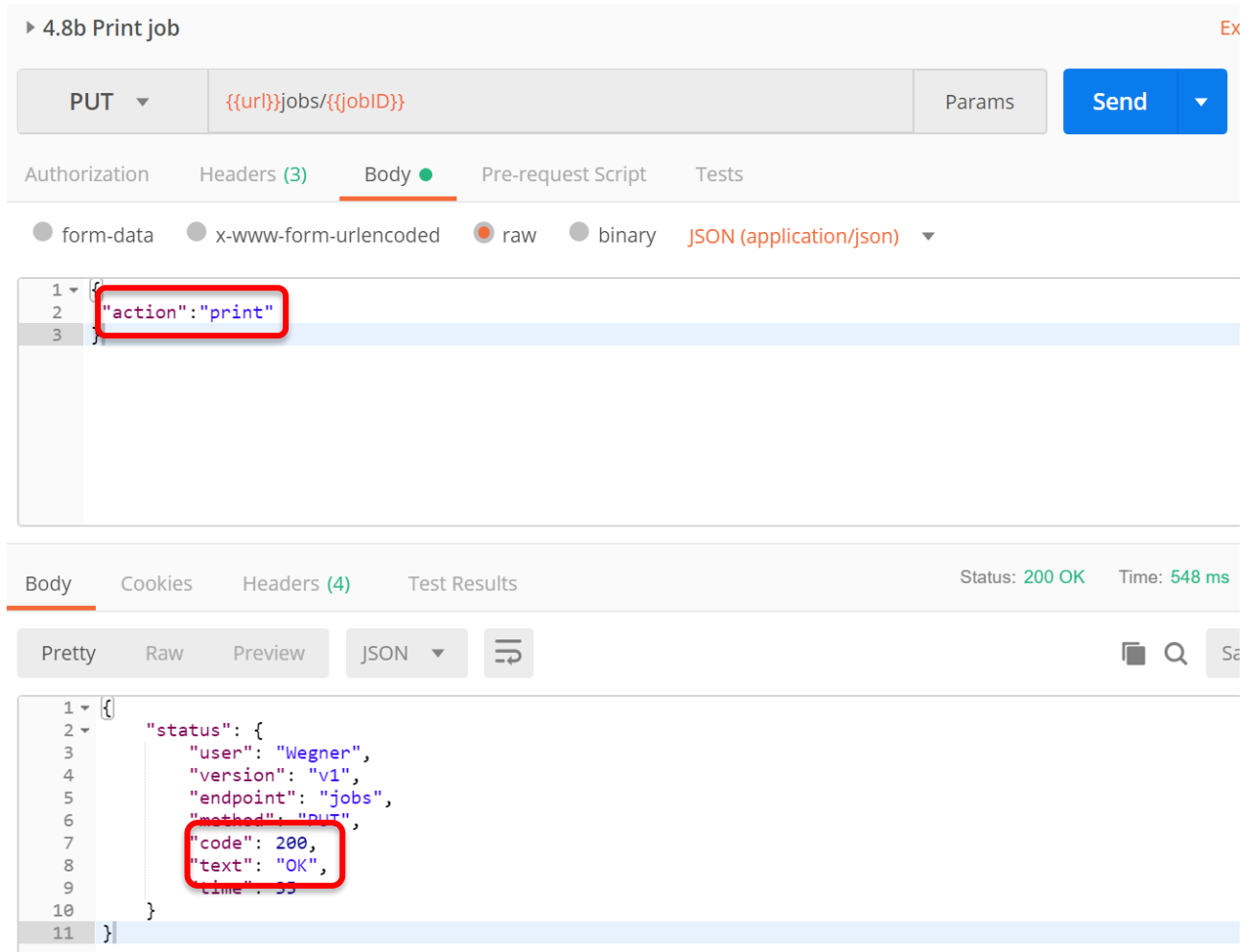


The screenshot displays an API client interface for a POST request to `{{url}}jobs`. The request body is a JSON object with `"hotfolder": "{{hotfolder}}"` and `"fileID": "{{fileID}}"`. The response status is 201 Created, and the response body is a JSON object containing a `"status"` object and a `"jobID"` string. The `"printed"` field in the response is set to `false`.

```
1 {
2   "hotfolder": "{{hotfolder}}",
3   "fileID": "{{fileID}}",
4 }
```

```
1 {
2   "status": {
3     "user": "Wegner",
4     "version": "v1",
5     "endpoint": "jobs",
6     "method": "POST",
7     "code": 201,
8     "text": "Created",
9     "time": 308
10  },
11  "jobID": "44dba491-416e-461e-9b3b-1a0ccbdb9cd8",
12  "jobName": "CMYK7.CSB",
13  "fileName": "cmyk.tif",
14  "size": "",
15  "copies": 1,
16  "fileSize": "0.00 MB",
17  "printed": false,
18  "backup": false
19 }
```

Send a request with parameter “action”:”print”. You should receive status OK which means printing has started.



The screenshot displays a REST client interface for a PUT request. The request is sent to the endpoint `{{url}}jobs/{{jobID}}` with the body `{ "action": "print" }`. The response is a JSON object with a status of 200 OK. The response body is shown in a pretty-printed JSON format.

```
PUT {{url}}jobs/{{jobID}} Params Send
```

Authorization Headers (3) **Body** Pre-request Script Tests

form-data x-www-form-urlencoded raw binary **JSON (application/json)**

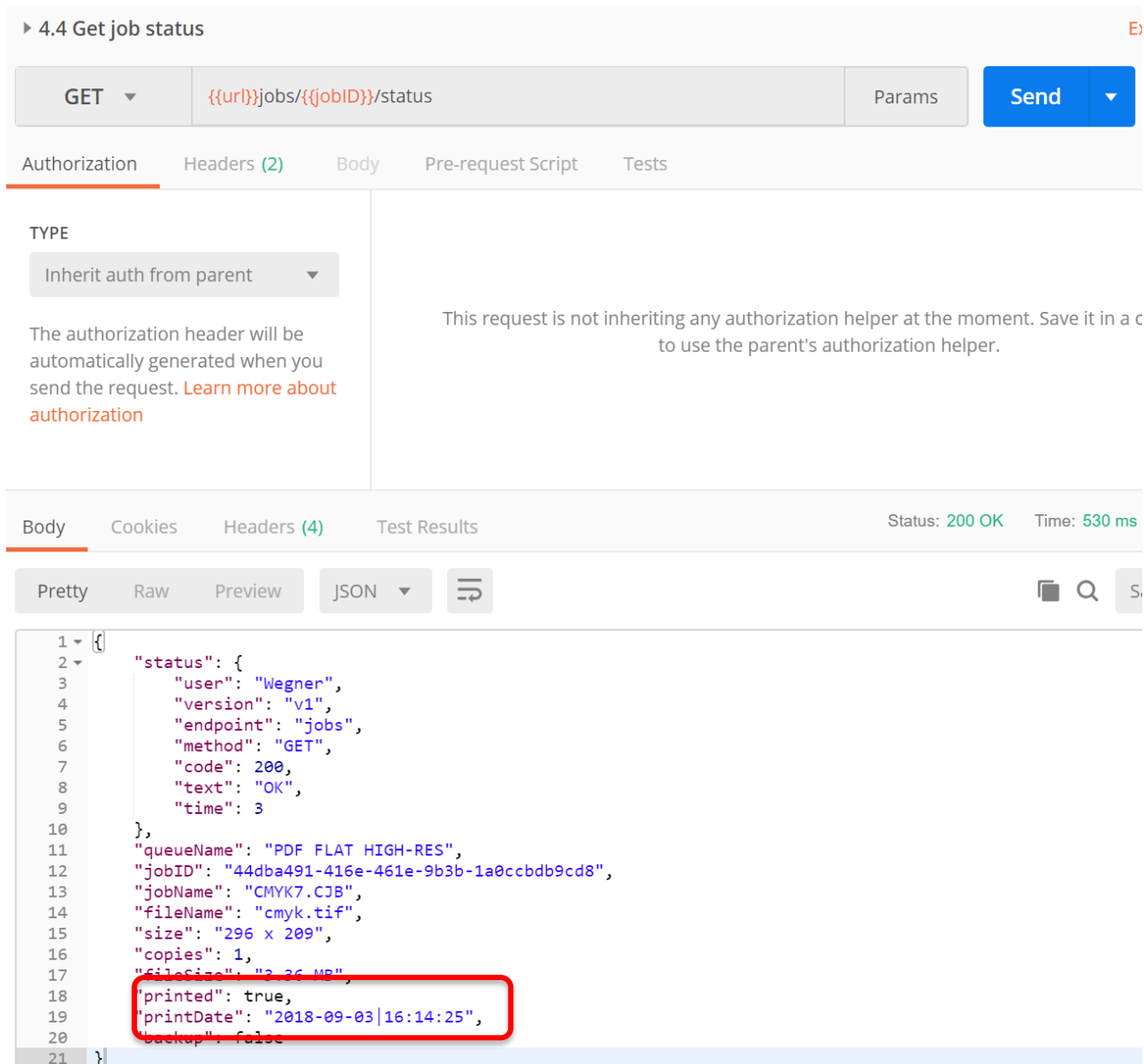
```
1 {
2   "action": "print"
3 }
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 548 ms

Pretty Raw Preview **JSON**

```
1 {
2   "status": {
3     "user": "Wegner",
4     "version": "v1",
5     "endpoint": "jobs",
6     "method": "PUT",
7     "code": 200,
8     "text": "OK",
9     "time": 35
10  }
11 }
```

Send a 4.4.1 request to receive the status of the job. Repeat the request once per second until the status “printed” is set to true.



▶ 4.4 Get job status

GET `{{url}}jobs/{{jobID}}/status` Params **Send**

Authorization Headers (2) Body Pre-request Script Tests

TYPE
Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a c to use the parent's authorization helper.

Body Cookies Headers (4) Test Results Status: 200 OK Time: 530 ms

Pretty Raw Preview JSON

```
1 {
2   "status": {
3     "user": "Wegner",
4     "version": "v1",
5     "endpoint": "jobs",
6     "method": "GET",
7     "code": 200,
8     "text": "OK",
9     "time": 3
10  },
11  "queueName": "PDF FLAT HIGH-RES",
12  "jobID": "44dba491-416e-461e-9b3b-1a0ccbdb9cd8",
13  "jobName": "CMYK7.CJB",
14  "fileName": "cmyk.tif",
15  "size": "296 x 209",
16  "copies": 1,
17  "fileSize": "3.26 MB",
18  "printed": true,
19  "printDate": "2018-09-03|16:14:25",
20  "backup": false
21 }
```