Tickets endpoint

Nov 10, 2025

All HappyFox help-desk instances are shipped with a RESTful web service API that enables various operations, including ticket creation, ticket update submission, ticket and user listing. This API supports payload formats such as JSON and multipart/form-data.

Requirements:

- Making HTTP requests (using GET and POST HTTP methods as a minimum requirement).
- Doing HTTP Basic Authentication.
- Generating and reading data in the JSON format.
- Optionally making HTTP POST requests using content type of "multipart/form-data" (needed for ticket attachments)

Documentation Conventions:

- The documentation indicates parameters that need to be replaced with actual values. The entire string including the enclosing < and > should be replaced. For example, if the parameter is **email**, it should be replaced with the required email address
- **Endpoint url format:** /api/1.1/json/<module>/ - This tail url has to be prefixed with your HappyFox account url.

  Eg:- https://acme.happyfox.com/api/1.1/json/<module>/
- All HTTP POST API calls are to be explicitly marked as **https**. Else, it would be considered as GET only.

> *Note:*
> *1) If you are using a custom domain, please use the custom domain url only.*
> *2) API key and auth code are to be passed in Basic HTTP authentication format only.*

- To update assignee, status, priority, ticket custom fields, user custom fields, category, etc in the payload please use their IDs. Refer this article (https://support.happyfox.com/kb/article/360-api-for-happyfox/) to fetch their IDs
- <module> - refers the module that is being accessed viz., /tickets/, /users/ etc. A full

constructed example url: https://acme.happyfox.com/api/1.1/json/tickets/

- If your HappyFox account is hosted in EU, please use <instance_name>.happyfox.net. Eg. acme.happyfox.net

List of topics covered in this article

Note: All timestamps in API responses are in UTC

1. Get a list of all Tickets

API Endpoint: /api/1.1/json/tickets/

Method: GET

Query parameters that can be passed to the url are listed below with their description

Param	Description
minify_response	true/false. If true, then show only list of ticket ids in the response
status	Filter tickets by status Values: *_all: return tickets in all statuses (default)* id of status: filter tickets by the given status
category	Filter tickets by category Values: * id of category: filter tickets by the given category
q	Used for the purpose of search and advanced filters. When using this, the url should be of the format: For pending behavior tickets - /api/1.1/json/tickets/?status=_pending&q="" For all tickets - /api/1.1/json/tickets/?status=_all&q=""
sort	sort field using advanced filters

Back to top

2. Get a paginated list of tickets

API Endpoint : /api/1.1/json/tickets/?size=<size value>&page=<page number>

Default ***size value*** is 10 and can be max 50

Method: GET

Query Parameters that can be passed to the url are listed below with their description

Filter fields

Field	Description
unresponded	true / false. if true, return unresponded tickets

breached	true / false. if true, return tickets that have breached at least one sla
has_attachments	true / false. if true, return tickets that have at least one attachment. This has to be passed as a search parameter: /api/1.1/json/tickets/?q=has_attachments:true
tag	Return tickets having the given tagsvalues format: comma separated values. Case sensitive.
duedate	allowed values: today - return tickets due today yesterday - return tickets that were due yesterday tomorrow - return tickets that are due tomorrow overdue - return tickets where due date is less than today next 7 days - return tickets due in 7 days
priority	--return tickets belonging to the given priorities --values format: comma separated names of priorities within quotations. Eg: /tickets/?status=_all&q=priority:"CRITICAL","MEDIUM"
status	---return tickets belonging to the given statuses --values format: comma separated names of statuses within quotations. Eg: /tickets/?status=_all&q=status:"New","Closed"
assignee	list of keywords separated by commas. allowed keywords: --none - unassigned tickets --any - return all assigned tickets --some keyword - return tickets assigned to agents whose email address / username / first name / last name match the keyword given (case sensitive, exact match)
group	list of keywords separated by commas. allowed keywords: --none - return tickets raised by contacts who do not belong to any contact group --any - return tickets raised by contacts who belong to at least one contact group --name of contact group - return tickets raised by contacts belonging to the contact group (case sensitive, exact match)
contact	--list of keywords separated by commas. each keyword is matched against a lookup generated for the contact (name, email, phone number) --return tickets associated to contacts who match any of the given keywords eg: /tickets/?status=_all&q=contact: “james@example.com”
id	list of ids of tickets separated by commas. we can search by id and display id. In case of display id, do not include the #. eg: q=id:DC00000001,2

category	Returns tickets of a specific category whose id is passed in the url. Eg / tickets/?status=_all&category=<id of the category>

2.1 Using url query parameters and search strings

Filter by time based fields (timestamp format: yyyy/mm/dd).

Search query format: /api/1.1/json/tickets/?status=_all&q=""

All of the following fields are to be passed as a search string as shown above.

Field	Description
created-on	get tickets created on a particular date eg: q=created-on:"2019/01/25"
created-on-or-before	get tickets created on or before a particular date eg: q=created-on-or-before:"2019/01/25"
created-on-or-after	get tickets created on or after a particular date eg: q=created-on-or-after:"2019/01/25"
created-before	get tickets created before a particular date eg: q=created-before:"2019/01/25"
created-after	get tickets created after a particular date eg: q=created-after:"2019/01/25"
last-modified-on-or-before	get tickets last modified on or before a particular date eg: q=last-modified-on-or-before:"2019/01/25"
last-modified-on-or-after	get tickets last modified on or after a particular date eg: q=last-modified-on-or-after:"2019/01/25"
last-modified-before	get tickets last modified before particular date eg: q=last-modified-on-or-after:"2019/01/25"
last-modified-after	get tickets last modified after particular date eg: q=last-modified-before:"2019/01/25"
last-staff-replied-on	get tickets where the latest staff reply was added on a particular date
last-staff-replied-on-or-before	get tickets where the latest staff reply was added on or before a particular date

last-staff-replied-on-or-after	get tickets where the latest staff reply was added on or after a particular date
last-staff-replied-before	get tickets where the latest staff reply was added before a particular date
last-staff-replied-after	get tickets where the latest staff reply was added after a particular date
last-contact-replied-on	get tickets where the latest contact reply was added on a particular date
last-contact-replied-on-or-before	get tickets where the latest contact reply was added on or before a particular date
last-contact-replied-on-or-after	get tickets where the latest contact reply was added on or after a particular date
last-contact-replied-before	get tickets where the latest contact reply was added before a particular date
last-contact-replied-after	get tickets where the latest contact reply was added after a particular date
last-closed-on-or-after	get tickets which were last closed on or after a particular date. Note that the current status of a ticket may be in pending. This only checks for tickets that were closed on or after the date.
last-closed-on-or-before	get tickets that were last closed on or before a particular date.

Filter by custom fields

We can search for tickets by contact and ticket custom fields

Format of specifying custom fields in search

- If the custom field name has only one word <name of the field>: “value” - do not enclose the field name in quotes
- If the custom field name has multiple words “<name of the field>”:”value” - enclose the field name in quotes “field_name”:”value”
- Custom field name and value should be enclosed in quotes separately.
- Name and value should be separated by a colon (:)
- No space after the colon separator
- Field names and values are not case sensitive
- Multiple fields should be separated by space

- For dropdown and multiple choice fields, the values are the option labels
- For multiple choice fields - each option is given in quotes and the options are separated by commas

Examples:

- text field: /tickets/?status=_all&q="Enterprise Company Name":"enterprise company"
- dropdown: /tickets/?status=_all&q="Team Name":"arsenal"
- multiple choice: /tickets/?status=_all&q="Country":"finland","germany"
- date field: /tickets/?status=_all&q="Date Field":"08/23/2019"
- number field: /tickets/?status=_all&q="Number Field":"300"
- multiple fields: /tickets/?status=_all&q=Field1:"value1" "Field2":"value 2"
- Filtering with contact and custom fields: /tickets/?status=_all&q=contact:"james@happyfox.com" "Product Name":"Helpdesk"
- To combine multiple search items for system fields, please build the search query by concatenating each search item with a +

  Eg. Search for status **In Progress** and **New**, along with tickets that contain that contain tag **routine_checks**

  To search statuses, the query string would be,
  /tickets/?status=_all&q=status:"In+Progress","New"

  To search with tags, the query string would be,
  /tickets/?status=_all&q=tag:"routine_checks"

  To combining the two,
  /tickets/?status=_all&q=tag:"routine_checks"+status:"In+Progress","New"

Sorting list of tickets

- Default sorting - last updated based on contact/agent reply in descending order.
- If we search tickets, the sort order is always by relevance and last updated descending.

Other sort options

- categorya - sort by category ascending

- categoryd - sort by category descending
- subjecta - sort by subject in alphabetical order ascending
- subjectd - sort by subject descending
- due - sort by due date
- statusd - sort by status order descending
- statusa - sort by status order ascending
- prioritya - sort by priority order ascending
- priorityd - sort by priority order descending
- updated - sort by last updated (agent / contact reply) descending
- updatea - sort by last updated (agent / contact reply) ascending
- unresponded - unresponded tickets first
- createa - sort by ticket creation date ascending
- created - sort by ticket creation date descending
- assigneea - sort by assignee username ascending
- assigneed - sort by assignee username descending
- last_modifieda - sort by ticket last modified date ascending
- last_modifiedd - sort by ticket last modified date descending
- ticketa - sort by ticket id ascending
- ticketd - sort by ticket id descending
- clienta - sort by contact id ascending
- clientd - sort by contact id ascending

Back to top

3. Ticket detail page

Endpoint - /api/1.1/json/ticket/<ticket_number>/

Note: Ticket ID is a unique identifier for each ticket. It is a combination of category prefix and ticket number. Eg. #HFS00000001. Here 1 is the ticket number and HFS is the category prefix. Ticket number and ticket ID can be obtained from the /tickets/ endpoint from the ***id*** and ***display_id*** attributes in each item of the ***data*** JSON array.

Ticket ID

Ticket number

Method: GET

Example Response

{
"status": {
"name": "In Progress",
"color": "0066CC",
"order": 2,
"default": false,
"behavior": "pending",
*"id**": 2**
},
"first_message": "Example message\n\n",
"last_user_reply_at": "2019-08-13 08:59:11",
"time_spent": 12,
"id": 3,
"messages_count": 2,
"subject": "Example ticket",
"category": {
"prepopulate_cc": "AR",
"description": "Default Category, please edit this.",
"time_spent_mandatory": false,
"public": true,
"id": 1,
"name": "Default Category"
},
"attachments_count": 1,
"last_updated_at": "2019-08-13 08:59:42",
"priority": {
"default": false,
"id": 3,
"name": "High",
"order": 3
},

"last_staff_reply_at": "2019-08-13 08:59:42",
"custom_fields": [
{
"compulsory_on_complete": false,
"name": "Product Color",
"value": "grey",
"value_id": null,
"type": "text",
"id": 1,
"visible_to_staff_only": false
}
],
"sla_breaches": 0,
"merged_tickets": [],
"visible_only_staff": null,
"display_id": "#DC00000003",
"due_date": "2019-08-16",
"tags": "test",
"last_modified": "2019-08-13 08:59:42",
"user": {
"name": "Welcome",
"primary_phone": {
"type": "o",
"number": "02323412",
"id": 1
},
"phones": [
{
"type": "o",
"number": "02323412",
"id": 1
}

```
],
"created_at": "2019-07-16 11:27:30",
"updated_at": "2019-08-13 08:59:11",
"pending_tickets_count": 3,
"contact_groups": [],
"tickets_count": 3,
"id": 1,
"email": "james@example.com",
"custom_fields": [
{
"name": "Company",
"value": "Changed company",
"value_id": null,
"type": "text",
"id": 2,
"visible_to_staff_only": false
}
]
},
"subscribers": [
{
"first_name": "Admin",
"last_name": "User",
"id": 1,
"email": "james@example.com"
}
],
"unresponded": false,
"created_at": "2019-08-13 08:59:11",
"source": "TKT_CREATION_ADMIN_PANEL",
"assigned_to": {
"name": "Admin User",
```

```
"is_account_admin": true,
"email": "james@example.com",
"role": {
"name": "Administrator",
"id": 1
},
"active": true,
"id": 1,
"categories": [
1
],
"permissions": [
"relate_tickets",
"manage_import_assets",
]
},
"updates": [
{
"priority_change": {
"new_name": "High",
"new": 3,
"old": 1,
"old_name": "Medium"
},
"category_change": null,
"custom_field_change": null,
"timestamp": "2019-08-13 08:59:42",
"due_date_change": {
"new": "2019-08-16",
"old": null
},
"by": {
```

"email": "james@example.com",
"type": "staff",
"id": 1,
"name": "admin"
},
"time_spent": 12,
"update_id": 52,
"message": {
"attachments": [
{
*"url": "*https://exampleurl.com/attachments/examplefile.jpg (https://exampleurl.com/attachments/examplefile.jpg)*",*
"id": 22,
"filename": "bicycle.gif"
}
],
"bcc_list": "",
"text": "Attachment added\n\n",
"cc_list": "james@example.com",
"customer_updated": false,
"html": "

",
"forward_list": null,
"message_type": null,
"subject": ""
},
"assignee_change": null,
"status_change": {
"new_name": "In Progress",
"new": 2,

```
"old": 1,
"old_name": "New"
},
"satisfaction_survey": null
}
],
"merged_to": null
}
```


To obtain a history of changes/updates made on ticket custom fields, add the below url parameter.

Endpoint: /api/1.1/json/ticket/<ticket_number>/?show_cf_changes=true
Method: GET
Example Response:

Without *show_cf_changes=true* parameter:

```
184            },
185            "satisfaction_survey": null
186        },
187        {
188            "priority_change": null,
189            "category_change": null,
190            "custom_field_change": null,
191            "timestamp": "2021-10-06 07:44:00",
192            "due_date_change": null,
193            "by": {
```


With *show_cf_changes=true* parameter:

```
        "priority_change": null,
        "category_change": null,
        "custom_field_change": [
            {
                "to": "Crawl Completed",
                "from": "",
                "id": "9",
                "name": "Crawl status"
            },
            {
                "to": "XYZ",
                "from": "",
                "id": "11",
                "name": "Area"
            },
            {
                "to": "Chennai",
                "from": "",
                "id": "3",
                "name": "City"
            }
        ],
```


Back to top

4. Create Ticket

Using this API endpoint, tickets can be created in all Public categories/Agents and Contacts visible categories

API Endpoint: /api/1.1/json/tickets/

Method:POST

Payload Fields

Field	description		Required

name	Name of the contact.		yes. Required for new contacts only. For existing contacts, the ID of the contact can be passed in the ***client*** payload field.
email	Email address of the contact		yes. Required for new contacts only. For existing contacts, the ID of the contact can be passed in the ***client*** payload field.
phone	Phone number of the contact as a string		no
subject	ticket subject		yes
text	ticket message in plain text format		text / html is required
html	ticket message in html		text / html is required
category	ID of the category in which the ticket should be created. (this has to be a public category)		yes
priority	--id of the priority of the ticket --defaults to the default priority		no
assignee	id of the agent to assign the ticket to. To leave the ticket unassigned, use "assignee" : null		no
tags	list of tags separated by commas.		no
cc	list of email addresses, separated by commas		no
bcc	list of email addresses, separated by commas		no

created_at	ticket creation time allowed formats: yyyy-mm-ddThh:mm:ss yyyy-mm-ddThh:mm:ss yyyy-mm-ddThh:mm:ss.ms yyyy-mm-ddThh:mm:ss.ms		no
due_date	allowed formats yyyy-mm-dd dd/mm/yyyy		no
attachments	*list of files to be uploaded* Size limit: the total size of all the files combined should not exceed 25 mb *Content type: In case attachments are given, the content type of the request should be multipart/form-data* File types: There are no restrictions currently regarding file types. *File encoding: If the file has an encoding associated with it, HappyFox will use that encoding when reading the file. If not it uses UTF-8 by default. There are no restrictions regarding the encoding of a file.*		no
visible_only_staff	true / false *specify whether the ticket is private or not*		no
c-cf-**<id of custom field>**	The <id of the custom field> is to be fetched from the list contact custom fields (/api/json/1.1/json/user_custom_fields/) api endpoint only. Do not use the IDs from the urls from the agent portal as they are different. Format Text field*: string* Number: integer / float with 2 decimal places Dropdown*: the id of the choice* Multiple choice: list of ids of options Date: yyyy-mm-dd		yes for mandatory custom fields

t-cf-**<id of custom field>**	The <id of the custom field> is to be fetched from the list custom fields api endpoint(/api/json/1.1/json/ ticket_custom_fields/) only. Do not use the IDs from the urls from the agent portal as they are different. Format Text field*: string* Number: integer / float with 2 decimal places Dropdown*: the id of the choice* Multiple choice: list of ids of options Date: yyyy-mm-dd		yes for compulsory fields

Example payload

Example 1: With custom fields

```
{
"name": "james",
"email": "james@example.com",
"category": 1,
"subject": "test ticket",
"text": "example ticket",
"t-cf-3": [1, 2],
"t-cf-1": "text field value",
"t-cf-5": 200,
"t-cf-4": "2019-12-20",
"c-cf-3": 1
}
```

Example 2: Creating ticket with attachments

Ticket with attachments can be created by passing the payload as multi-part/form type. Here is a sample code in Python.

```
import requests
url = "https://<account_name>.happyfox.com/api/1.1/json/tickets/"
payload={'category': '4',
'name': 'Name',
'text': 'Ticket creation message',
'subject': 'Ticket creation subject',
'email': 'john@example.com',
't-cf-5': '1',
't-cf-25': '1'}
files=[
('attachments',('Second_screen.png',open('/Users/Laura/Desktop/Second_screen.png','rb'),'image/
png'))
]
headers = {
'Authorization': 'Basic <BASIC_AUTH_STRING>'
}

response = requests.request("POST", url, headers=headers, data=payload, files=files)

print(response.text)

# To generate the BASIC_AUTH_STRING, please use the below code snippet
#from requests.auth import _basic_auth_str
#v1_api_key = <API_Key>
#v1_auth_code = <AUTH_Code>
#v1_auth_token = _basic_auth_str(v1_api_key, v1_auth_code)
```

Successful response data for status 200 is as same as /ticket/

Example failure response (400)

```
{
"error": [
    {
        "field": "category",
        "errors": [
            "This field is required."
        ]
    }
]
}
{
"error": [
    {
        "field": "t-cf-3",
        "errors": [
            "This field is required"
        ]
    }
]
}
```


Back to top

5. Create multiple tickets

API Endpoint: /api/1.1/json/tickets/

Method:POST

Payload

- The payload should be a list of ticket payloads.
- For the payload structure of one ticket refer the Create Ticket section
- A maximum of 100 tickets can be created in one request
- Response will be a list of dictionary, indicating successfully created tickets and payloads with validation errors (ref example below)
- success - false indicates that the payload had validation errors
- success - true indicates that the ticket got created successfully

Example response

```
[
{
    "display_id": "#DC00000011",
    "id": 11,
    "success": true
},
{
    "success": false,
    "error": [
        {
            "field": "category",
            "errors": [
                "This field is required."
            ]
        }
    ]
}
]
```

Back to top

6. Create an inline attachment to be added to the ticket body

- Use the below endpoint to upload the inline attachment to HappyFox.
- This will generate a temporary url
- Set this temporary url as the value of src property of img tag in the ticket message (html)

when creating a ticket

API Endpoint : /api/1.1/json/ticket-inline-attachment

Method: POST

Payload Fields

Field	description	Required
file	*the file to be uploaded. Only one file can be upload at a time* Size limit: should not exceed 25 mb *Content type: content type of the request should be multipart/form-data* File types: Should be an image file * File encoding: If the file has an encoding associated with it, HappyFox will use that encoding when reading the file. If not it uses utf-8 by default. There are no restrictions regarding the encoding of a file.	yes

Example success response (200)

```
{
  "url": "https://example.happyfox.com/get_hdp_temporarily_attachment/2091/"
}
```

Please reach out to support@happyfox.com for an example code snippet (mention the coding language) that allows you to create ticket with inline and external attachments.

Back to top

7. Fetch ticket attachment urls

API Endpoints:
/api/1.1/json/ticket/<ticket_number>/
/api/1.1/json/tickets/

Method: GET

The api response for these endpoints contain a field called attachments_count that contains the total count of attachments in the ticket.

```
{
...

...
...
"attachments_count": 1,
"last_updated_at": "2025-07-18 08:09:37",
"priority": {
"default": true,
"id": 5,
"name": "No Priority",
"order": 1
...
...
...

}
```

The attachment urls are listed in the attachments array for every element present in the updates array for each ticket element in the data field. Each url has an expiry time of 5 mins. The attachment can be accessed directly using the link as the authentication code is embedded in the url for one time use.

Note: attachments, updates and data are payload fields in the api response.

```
{
...
...
...
"message": {
"attachments": [
{
"url": "https://hf-files-oregon.s3.amazonaws.com/
hdpdemofox_email_attachments/2021/07-19/426cd298-63c8-4ff8-8161-93e19545df48/
home-2741413_1280_1.png?
Signature=jdvU%2Bzx6tQaf%2BdjnkZwi45iRoo0%3D&Expires=1752826500&AWSAccessKeyId=A
KIASBRBREIGNJFFU4OL",
"id": 321,
"filename": "home-2741413_1280 (1).png"
}
],
...
...
...
}
```


8. Adding a Staff update

API Endpoint: /api/1.1/json/ticket/<ticket_number>/staff_update/

Method: POST

Payload Fields

Field	description	Required
staff	ID of the agent who is adding the reply	yes

cc	list of email addresses separated by commas	no
bcc	list of email addresses separated by commas	no
update_customer	• true / false. • indicate whether to send the reply notification to the ticket contact • defaults to false	no
send_survey	• true / false • indicate whether to launch survey • defaults to false	no
parent_update	• indicate the parent update in tickets created for facebook / twitter conversations	no
last_staff_message	• the id of the last staff message (get this id from ticket -> updates -> message) • this is used to specify an update which will be checked against the ticket's last staff message for the purpose of agent collision	no
subject	change reply subject to be sent to the contact	no
status	ID of the status. Changes status of the ticket	no
priority	ID of the priority. Changes priority of the ticket	no
assignee	ID of the assignee. Changes assignee of the ticket. To leave the ticket unassigned, use "assignee" : null	no
time_spent	Integer. Adds time spent (in minutes) to the ticket	depends on category settings
due_date	allowed formats • yyyy-mm-dd • dd/mm/yyyy	no
tags	list of tags separated by commas	no

attachments	• list of files • total size of all attachments should not exceed 25 mb	no
html	agent reply in html format	no
plaintext	agent reply in plain text format	no
ccf-<id of custom field>	Contact custom fields Get the id of the custom fields from list contact custom fields api response values format: • text field: string • number: integer / float with 2 decimal places • dropdown: the id of the choice • multiple choice: list of ids of options • date: yyyy-mm-dd	no
t-cf-<id of custom field>	Ticket custom fields Get the id of the custom fields from list contact custom fields api response values format: • text field: string • number: integer / float with 2 decimal places • dropdown: the id of the choice • multiple choice: list of ids of options • date: yyyy-mm-dd	yes for compulsory on complete fields if the status is moved from pending to closed status.

Example payload

{
"staff": 1,
"status": 4,
"html": "
Example reply
",

"t-cf-3": [1, 2],
"c-cf-3": 1
}

Example success response

Refer ticket details example response

Example failure response

{
"error": {

```
"t-cf-2": "This field should be filled before marking this ticket as completed"
```

}
}

Note: Concurrent/parallel API calls to this endpoint, for any given particular ticket, is not supported.

Back to top

9. Adding a Staff private note

API Endpoint: /api/1.1/json/ticket/<ticket_number>/staff_pvtnote/

Method: POST

Payload Fields

Field	description	Required
staff	ID of the agent adding the reply	yes
alert	send private not alert to *s - all ticket subscribers* c - all agents associated to ticket's category *ID of the agent who should be alerted*	no
status	ID of the status. Changes status of the ticket	no
priority	ID of the priority. Changes priority of the ticket	no
assignee	ID of the assignee. Changes assignee of the ticket. To leave the ticket unassigned, use "assignee" : null	no
time_spent	integer. Adds time spent (in minutes) to the ticket	depends on category settings
due_date	allowed formats yyyy-mm-dd *dd/mm/yyyy*	no
tags	list of tags separated by commas	no
attachments	list of files *total size of all attachments should not exceed 25 MB*	no
html	agent's private note in html format	no
plaintext	agent private note in plain text format	no
ccf-**<id of custom field>**	Contact custom fields Get the id of the custom fields from list contact custom fields api response Values format: text field: string *number: integer / float with 2 decimal places* dropdown: the id of the choice *multiple choice: list of ids of options* date: yyyy-mm-dd	no
t-cf-**<id of custom field>**	Ticket custom fields Get the id of the custom fields from list contact custom fields api response Values format: *text field: string* number: integer / float with 2 decimal places *dropdown: the id of the choice* multiple choice: list of ids of options * date: yyyy-mm-dd	yes for compulsory on ticket completion fields if the status is moved to closed status.

Back to top

10. Adding a Contact reply

API Endpoint: </api/1.1/json/ticket/<ticket_number>/user_reply/

Method: POST

Sample payload:

```
{

"text": "This is a response sent by the user via API",
"cc": "exampleaddress@example.com",
"bcc" : ""exampleaddress-2@example.com",
"user": 27662
}
```

Payload Fields

Field	description	Required
user	**ID of the contact who is adding the reply**	**yes**
cc	list of email addresses separated by commas	no
bcc	list of email addresses separated by commas	no
attachments	*list of files* total size of all attachments should not exceed 25 MB	no
text	Reply message in text format or html format	yes

Back to top

11.Edit custom fields values of a ticket

API Endpoint:

/api/1.1/json/ticket/<ticket_number>/update_custom_fields/
Method: POST

Payload Fields

Field	description	Required
staff	ID of the agent	yes
t-cf-<id>	Ticket custom fields. Get the id of the custom fields from list contact custom fields api response values format: *text field: string* number: integer / float with 2 decimal places *dropdown: the id of the choice* multiple choice: list of ids of options * date: yyyy-mm-dd	yes for compulsory fields

Back to top

12. Update tags of a ticket

API Endpoint: /api/1.1/json/ticket/<ticket_number>/update_tags/

Method: POST

Payload Fields

Field	description	Required
add	list of tags to add separated by commas	no
remove	list of tags to remove separated by commas	no

staff_id	ID of the agent making the update. To get the ID of the agents, please check /api/1.1/json/staff/	yes, to show the agent name in the ticket

Example Request

```
{
"add": "test1, example",
"remove": "test",
"staff_id" : 1
}
```

For a sample success response please refer ticket details response

Back to top

13. Subscribe to ticket (agent only)

API Endpoint: /api/1.1/json/ticket/ticket_number/subscribe/

Method: POST

Payload Fields

Field	description	Required
staff_id	ID of the agent to be added as a subscriber to ticket. This can be fetched from /api/1.1/json/staff/	yes
data	list of ids of agents to be added as subscribers to ticket	no

Example Request

```
{
"data": [3, 2],
"staff_id": 1
}
```

Example success response

```
{
"message": "Admin Agent, james may, Subscribers added to ticket"
}
```

Back to top

14. Unsubscribe from ticket (agent only)

API Endpoint: >/api/1.1/json/ticket/<ticket_number>/unsubscribe/

Method: POST

Payload Fields

Field	description	Required
staff_id	ID of the agent to be added as a subscriber to ticket - should be “ID of the agent to unsubscribe from a ticket”. This can be fetched from /api/1.1/json/staff/	yes

Example Request

```
{
"staff_id": 1
}
```

Example for a successful response

```
{
"message": "Admin Agent unsubscribed"
}
```

Back to top

15. Forward ticket to an external email address

API Endpoint : /api/1.1/json/ticket/<ticket_number>/forward/

Method: POST

Payload Fields

Field	description	Required
staff_id	ID of the agent who is forwarding the ticket. This can be fetched from /api/1.1/json/staff/	yes
to_include_ticket_contact	true / false	true / false
cc_include_ticket_contact	true / false	no
to	list of email addresses separated by commas	yes
cc	list of email addresses separated by commas	no
bcc	list of email addresses separated by commas	no
subject	email subject	yes
message	email message	yes
send_all_messages	true / false	no. defaults to true
include_pvt_notes	true / false	no. default value is false
ticket_attachments	*list of ids of ticket attachments.* Get the ticket attachment ids using get ticket details API endpoint	no

attachments	*list of files* total size of all the attached files should not exceed 25 MB	no
convert_replies_as_new_ticket	true / false	no. default value is true

Example Request

```
{
"staff_id": 1,
"subject": "test",
"to": "james@example.com, john@example.com"
}
```

Example successful response

```
{
"message": "Successfully forwarded the ticket #DC00000004"
}
```

Back to top

16. Move ticket to another Category

API Endpoint : /api/1.1/json/ticket/<ticket_number>/move/

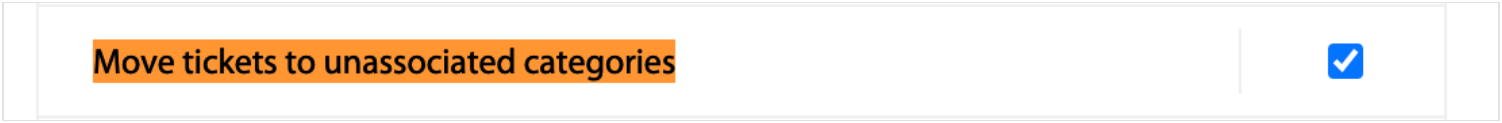
POST

Payload Fields

Field	description	Required
staff_id	ID of the agent who is changing the ticket category. This can be fetched from /api/1.1/json/staff/	yes
target_category_id	ID of the category to move the ticket to	yes
move_note	Note to be added when moving the ticket	no

assign_to	ID of the agent to whom the ticket has to be assigned to	no

Note: **The staff used for the above action should have the below permission enabled for their role.**



Example Request

```
{
"staff_id": 1,
"target_category_id": 2,
"move_note": " move note text ",
"assign_to": 2
}
```

Example success response

```
{
"status_code": 200,
"message": "moved ticket to Test"
}
```

Back to top

17. Delete ticket

API Endpoint : /api/1.1/json/ticket/<ticket_number>/delete/

Method: POST

Payload Fields

Field	description	Required
staff_id	ID of the agent that is performing the category change. This can be fetched from / api/1.1/json/staff/	yes

Example request:

```
{
"staff_id": 1
}
```

Example successful response:

```
{
"deleted_ticket": "#DC00000005"
}
```

Back to top

18. Create ticket with attachments

Sample python script to create a new ticket with attachments is available here (https://support.happyfox.com/kb/article/222/)

Note: Ticket creation with attachments is supported only for HTTP POST, with multi-part/form data payloads.

19. List all ticket custom fields

API Endpoint: /api/1.1/json/ticket_custom_fields/

Method: GET

Example response:

```
[
{
"name": "City",
"depends_on_choice": null,
"required": false,
"compulsory_on_completed": true,
"choices": null,
"compulsory_on_move": false,
"type": "text",
"id": 1,
"categories": [
{
"category": 1,
"order": 1
}
],
"visible_to_staff_only": false
},
{
"name": "State",
"depends_on_choice": null,
"required": false,
"compulsory_on_completed": false,
"choices": null,
"compulsory_on_move": false,
"type": "text",
"id": 2,
"categories": [
{
"category": 1,
"order": 2
```

```
},
{
"category": 2,
"order": 1
},
{
"category": 3,
"order": 1
},
{
"category": 4,
"order": 1
},
{
"category": 5,
"order": 1
}
],
"visible_to_staff_only": false
},
{
"name": "City",
"depends_on_choice": null,
"required": false,
"compulsory_on_completed": false,
"choices": [
{
"text": "Bengaluru",
"id": 4,
"dependant_fields": []
},
{
```

"text": "Chennai",
"id": 1,
"dependant_fields": []
},
{
"text": "Hyderabad",
"id": 5,
"dependant_fields": []
},
{
"text": "Kochi",
"id": 7,
"dependant_fields": []
},
{
"text": "Mumbai",
"id": 3,
"dependant_fields": []
},
{
"text": "New Delhi",
"id": 2,
"dependant_fields": []
},
{
"text": "Trivandrum",
"id": 6,
"dependant_fields": []
}
],
"compulsory_on_move": false,
"type": "choice",

```
"id": 3,
"categories": [
{
"category": 1,
"order": 3
},
{
"category": 2,
"order": 2
},
{
"category": 3,
"order": 2
},
{
"category": 4,
"order": 2
},
{
"category": 5,
"order": 2
}
],
"visible_to_staff_only": false
}
]
```


Back to top

20. Fetch specific JSON fields

Specific JSON fields from the top-level of the JSON structure can be retrieved using this method. The

names of the fields are passed as comma separated values to the ***fields*** parameter.

API Endpoint: /api/1.1/json/tickets/?fields=last_user_reply_at,last_staff_reply_at&size=50

HTTP Method: GET

Example: /api/json/1.1/tickets/?fields=id,last_user_reply_at,last_staff_reply_at

Example response:


```
{
    "page_info": {
        "count": 50,
        "last_index": 440,
        "page_count": 9,
        "start_index": 1,
        "end_index": 50
    },
    "data": [
        {
            "last_staff_reply_at": null,
            "id": 5392,
            "last_user_reply_at": "2025-07-15 17:39:35"
        },
        {
            "last_staff_reply_at": "2025-07-15 09:38:39",
            "id": 5391,
            "last_user_reply_at": "2025-07-10 16:29:48"
        },
        {
            "last_staff_reply_at": null,
            "id": 5390,
            "last_user_reply_at": "2025-07-10 16:25:18"
        },

        ...

        ...

        ...
    ]
}
```