Exporting Data

Exporting Conversations via API

GET /conversation/export

`/conversation/export` can be used to regularly export all conversation data from Decagon. This endpoint returns up to 100 conversations at a time, along with all messages, customer ratings, tags, and other metadata associated with each of those conversations.

Parameters:

All parameters below are optional!

1. **cursor:** To get subsequent pages of conversations, pass in the value returned as `next_page_cursor` from a prior call. Leave this blank to get the oldest conversations.
2. **min_timestamp:** The minimum timestamp at which a conversation was last updated. Note that this should be an integer representing the number of seconds since 1970-01-01. In Python, you can get a timestamp by doing `datetime.now().timestamp()` .
3. **max_timestamp:** The maximum timestamp at which a conversation was last updated. Note that this should be an integer representing the number of seconds since 1970-01-01. In Python, you can get a timestamp by doing `datetime.now().timestamp()` .
4. **user_filters:** Filters to apply to the export, which is stringified JSON. Currently the only filter supported is the metadata filter. To retrieve all records from a user with a specific email set in metadata, use this filter `{"metadata_filter": {"email": "jane@doe.com"}}` .
5. **use_updated_at:** Specify whether timestamps apply to the `updated_at` field for conversations or the `created_at` field. This field is `False` by default, thus `created_at` . Set to `True` to use `updated_at` .

You might often get duplicate conversation even if you query with min and max timesta ps that don't overlap. Let's say it is currently Jan 1 at 10am, and you query for all conversations from Dec 31 at 10am to Jan 1 at 10am. Now, tomorrow (Jan 2 at 10am) you query for all conversations from Jan 1 at 10am to Jan 2 at 10am — this could return some conversations that were returned the prior day as well, if those conversations had new messages added to them in the intervening 24 hour period!

Make sure you handle updating duplicates correctly!

Returns:

1. **conversations:** An array of conversations, along with all messages, customer ratings, tags, and other metadata associated with each of those conversations.
2. **next_page_cursor:** A value that can be passed in to a future call to this endpoint to get the next page of conversations. This value will be None if there are no more conversations to return.

Example Response:

```
"conversations": [{
    "conversation_id": "8ba9020c-0424-4bb2-ba5f-971a522c84de",
    "user_id": "user_12",
    "destination": "AI",
    "created_at": "2024-01-01 21:42:10.309970",
    "updated_at": "2024-01-01 21:49:25.309970",
    "last_user_message_at": "2024-01-01 21:45:18.309970",
    "metadata": {
        "user_type": "VIP",
    },
    "undeflected": true,
    "summary": "The customer reached out to say hello.",
    "resolution": "The agent offered to help the customer.",
    "messages": [{
        "text": "Hi there!",
        "role": "USER",
        "created_at": "2024-01-01 21:42:10.309970",
    }, {
        "text": "Hello! How can I help you?",
        "role": "AI",
        "created_at": "2024-01-01 21:42:12.309970",
    }],
    "flow_type": "CHATBOT",
    "csat_rating": 4,
    "csat_role": "AGENT",
    "tags": [{
        "name": "Greeting",
        "level": 0,
    }],
    "notes": "This is an internal note.",
    "audit_logs": [{
        "text": "CSAT set to 4",
        "payload": {},
        "created_at": "2024-05-06T14:11:26.799976",
    }],
}, {
    ...
}],
```

```
  "next_page_updated_after": 151050148,
```

flow_type is one of `CHATBOT` , `EMAIL` , `VOICE` , `AGENT_ASSIST` , `SMS` , `FORUM`

Conversations will be “updated” (and hence show up in future calls to this endpoint) if they receive new messages. Make sure to deduplicate in your code appropriately.

Example code:

Here is some example code that could be used to export all conversations from Decagon:

```python
import requests
import json

# Set up constants
API_KEY = "YOUR_API_KEY"
ENDPOINT = "https://api.decagon.ai/conversation/export"
HEADERS = {
    "Authorization": f"Bearer {API_KEY}"
}

# Either set cursor to 0, or set it to be the latest value
# of cursor from a prior call
cursor = 0

while True:
    params = {
        "cursor": cursor
    }

    # Make request and parse response
    response = requests.get(ENDPOINT, headers=HEADERS, params=params)
    response.raise_for_status()
    response = response.json()

    # Use response["conversations"] and save results
    pass

    # Update cursor
    cursor = response['next_cursor']
    if not cursor:
        break
```

File API

Overview