SecureMsgX documentations

Introduction:

Welcome to **SecureMsgX** – your trusted platform for sending and managing encrypted messages with the highest level of security and privacy. SecureMsgX offers a unique messaging experience where every message, or "scroll," is protected by advanced encryption, strict access controls, and limited view counts. Whether you want to send a one-time private note, a self-destructing secret, or hold a secure group conversation, SecureMsgX has you covered. This documentation will guide you through all the features, types of messages, and how to use the system effectively, whether you're a casual user looking to send secure messages or a developer integrating with our API. Get ready to communicate with confidence, knowing your messages remain private, secure, and under your control.

Why Security Wizards Choose SecureMsgX:

🔐 **End-to-End Encryption:** All messages are encrypted using strong algorithms (AES-256, ChaCha20, or Twofish). Only the sender and recipient can read them.

📜 **Flexible Scroll Types:** Choose from one-time messages, self-destructing notes, group chats, or broadcasts — depending on your purpose.

🧨 **Self-Destructing Messages:** Control how many times a message can be viewed — from just once to a billion views.

🙈 **No Unwanted Replies:** Scroll types like SINGLE, SECURE_SINGLE, and BROADCAST prevent replies – ideal for announcements or private notes.

👥 **1-to-1 and Group Conversations:** Use THREAD and GROUP types for secure conversations with individuals or multiple participants.

🛡️ **Zero Server Visibility:** Messages are encrypted client-side, meaning even the server cannot read or store your content.

📊 **API-Ready:** For technical users or teams, SecureMsgX provides well-structured API endpoints to automate or integrate message flows.

🚀 **Lightweight and Fast:** Built with performance in mind using Spring Boot, PostgreSQL, and Docker — ensuring a smooth, responsive experience.

🌍 **Globally Accessible:** Deployed on AWS infrastructure, with a containerised backend — scalable and reliable from anywhere.

🧨 *Scroll Categories and Access Rules:*

Scroll Type	Replies Allowed	Max Views	Description
SINGLE	❌ No	5 views (fixed)	One-time private message. Cannot be replied to
SECURE_SINGLE	❌ No	1 view (fixed)	Extremely sensitive message. Self-destructs after a single read.
BROADCAST	❌ No	1 to1,000,000,000 (custom set)	Announcement to many. Sender defines how many views are allowed.
THREAD	✅ Yes	Custom (per scroll)	Private 1-to-1 conversation. Unlimited replies allowed.
GROUP	✅ Yes	Custom (per scroll)	Secure group discussion. Unlimited replies allowed from participants.

📝 **Note:** Each scroll (ticket) type can be easily identified by the prefix in its `ticket_number`. This prefix helps distinguish the type of message at a glance:

- `SGL-` → SINGLE
- `SSL-` → SECURE_SINGLE
- `BRC-` → BROADCAST
- `THD-` → THREAD
- `GRP-` → GROUP

Example: `BRC-0000019834af0d3cd3c7b79d97eb7f4c55d3`

🔥 *Available API Endpoints (Doors of Durins)*

Below is a list of available SecureMsgX API endpoints—known as the **Doors of Durins –** each serving a specific purpose in scroll (ticket) creation, viewing, replying, deletion, and system monitoring. In the sections that follow, we'll explore each endpoint in detail with real-world examples, demonstrating how to send requests and interpret responses effectively.

Method	Endpoints	Purpose	Summary Description
POST	/doors-of-durin/sigil-scrolls/new-ticket	🎨 Create Ticket	Creates a new encrypted scroll (ticket).
POST	/doors-of-durin/sigil-scrolls/view	🔍 View Ticket Content	Views a scroll's content, applying view rules.
POST	/doors-of-durin/sigil-scrolls/replies	💬 Post Reply	Views a scroll's content, applying view rules.
DELETE	/doors-of-durin/sigil-scrolls/delete/{ticketId}	🗑️ Delete Ticket	Permanently deletes a scroll by ID.
GET	/doors-of-durin/sigil-scrolls/api-usage-metrics	📊 API Usage Metrics	Returns ticket creation and view stats.

📝 ***Note:*** Only the original sender of a scroll is authorized to permanently delete it, and this action must be performed using the internal `ticketId`. The `ticketNumber` (e.g., `SGL-000001...`) is used for referencing, viewing, and replying to scrolls, but **cannot** be used for deletion operations.

🔐 ***Supported Encryption Algorithms:*** supports multiple industry-grade encryption algorithms to protect message content. Each scroll is encrypted client-side using one of the following schemes.

Algorithm Name	Cipher Transformation	Key Length	Engine Used
AES_256	AES/GCM/NoPadding	256 bits (32 bytes)	AES
CHACHA20	ChaCha20-Poly1305	256 bits (32 bytes)	ChaCha20
TWOFISH	Twofish/GCM/NoPadding	256 bits (32 bytes)	Twofish

📝 ***Note:*** When creating a ticket, you must specify the encryption algorithm using the exact enum value in the request body: example: "encryption_algo": "AES_256"

🧪 ***Real-World Usage of Ticket Types:***
Now that we understand the different `scroll` behaviors and the available API endpoints, let's explore how each ticket type works in practice.

✅ ***How to create the ticket*** *(via `/new-ticket`):* To create a scroll (ticket), send a `POST` request to the `/new-ticket` endpoint with the required payload. Here is an example for creating a **THREAD** type scroll.

Endpoint: POST http://3.91.186.101:8083/doors-of-durin/sigil-scrolls/new-ticket
Header: Content-Type: application/json
Payload:


```
{
 "message_content": "This is an encrypted message.",
 "encryption_algo": "AES_256",
 "passkeys": [
  "k3y33",
  "k9y"
 ],
 "salt": "abc",
 "expires_at": "2025-07-30T23:59:59Z",
 "open_from": "",
 "open_until": "",
 "ticket_type": "THREAD",
 "max_views": 10
}
```

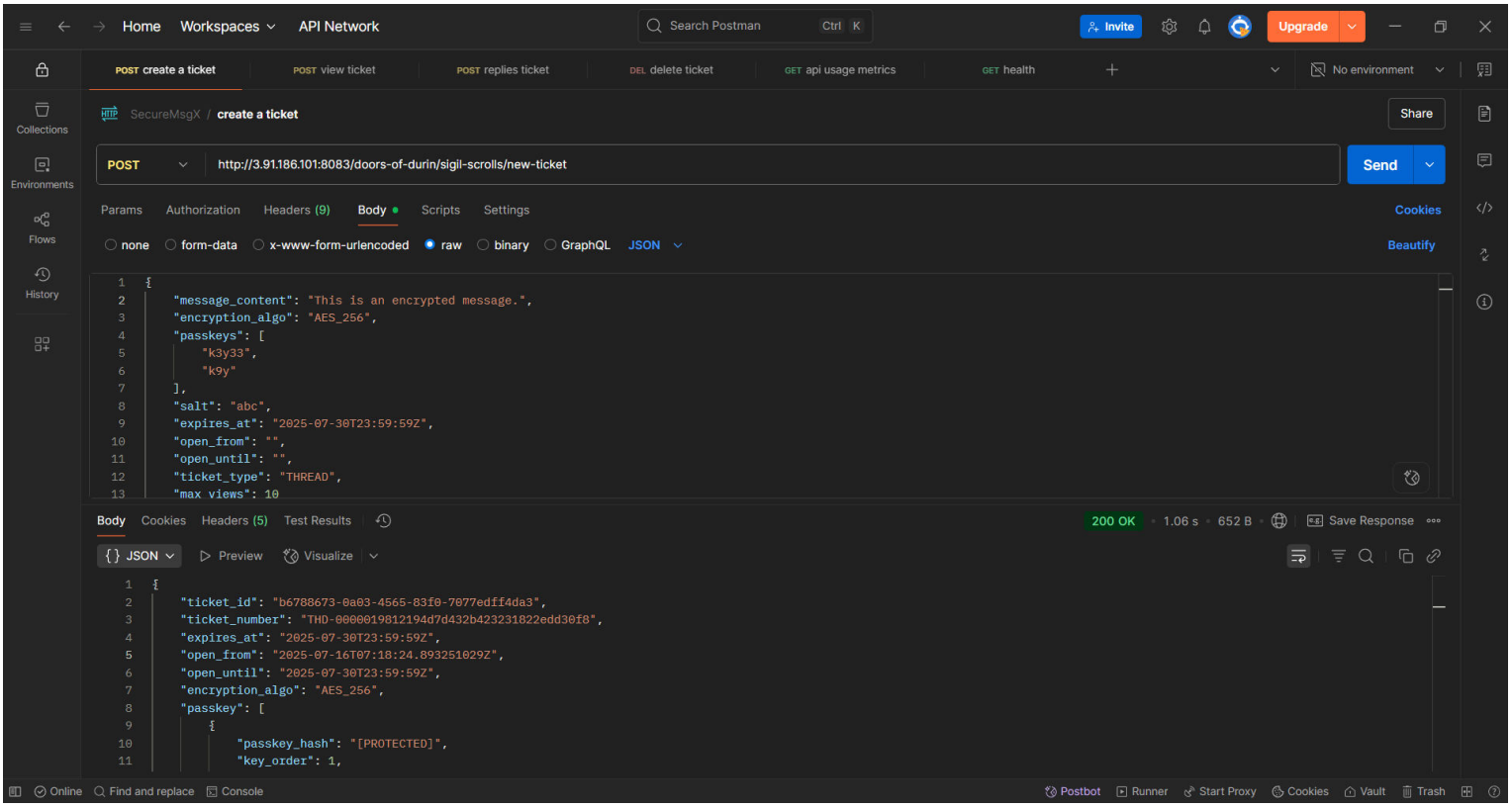

Response:


```
{
   "ticket_id": "b6788673-0a03-4565-83f0-7077edff4da3",
   "ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
   "expires_at": "2025-07-30T23:59:59Z",
   "open_from": "2025-07-16T07:18:24.893251029Z",
   "open_until": "2025-07-30T23:59:59Z",
   "encryption_algo": "AES_256",
   "passkey": [
      {
         "passkey_hash": "[PROTECTED]",
         "key_order": 1,
         "passkey": "k3y33"
      },
      {
         "passkey_hash": "[PROTECTED]",
         "key_order": 2,
         "passkey": "k9y"
      }
   ],
   "salt": "abc",
   "ticket_status": "OPEN",
   "ticket_type": "THREAD",
   "allow_replies": true,
   "count_views": 0
}
```

📝 ***Note: Scroll Access Timing Rules:*** When creating a scroll, you must choose one of the following access timing options:

- **Option 1: `expires_at`**
  Define a simple expiration timestamp after which the scroll becomes inaccessible.
- **Option 2: `open_from` and `open_until`**
  Set a custom access window during which the scroll can be viewed.

🚫 **Do not provide both** `expires_at` and `open_from`/`open_until` in the same request. Only one approach is allowed per scroll, and using both will result in a validation error.



🔍 ***How to view the ticket (via /view):*** Scroll viewing allows recipients to securely access the encrypted message content using the correct passkeys. Every scroll—regardless of type—can be viewed securely within its allowed view count and access window.

Endpoint: POST http://3.91.186.101:8083/doors-of-durin/sigil-scrolls/view

Header: Content-Type: application/json

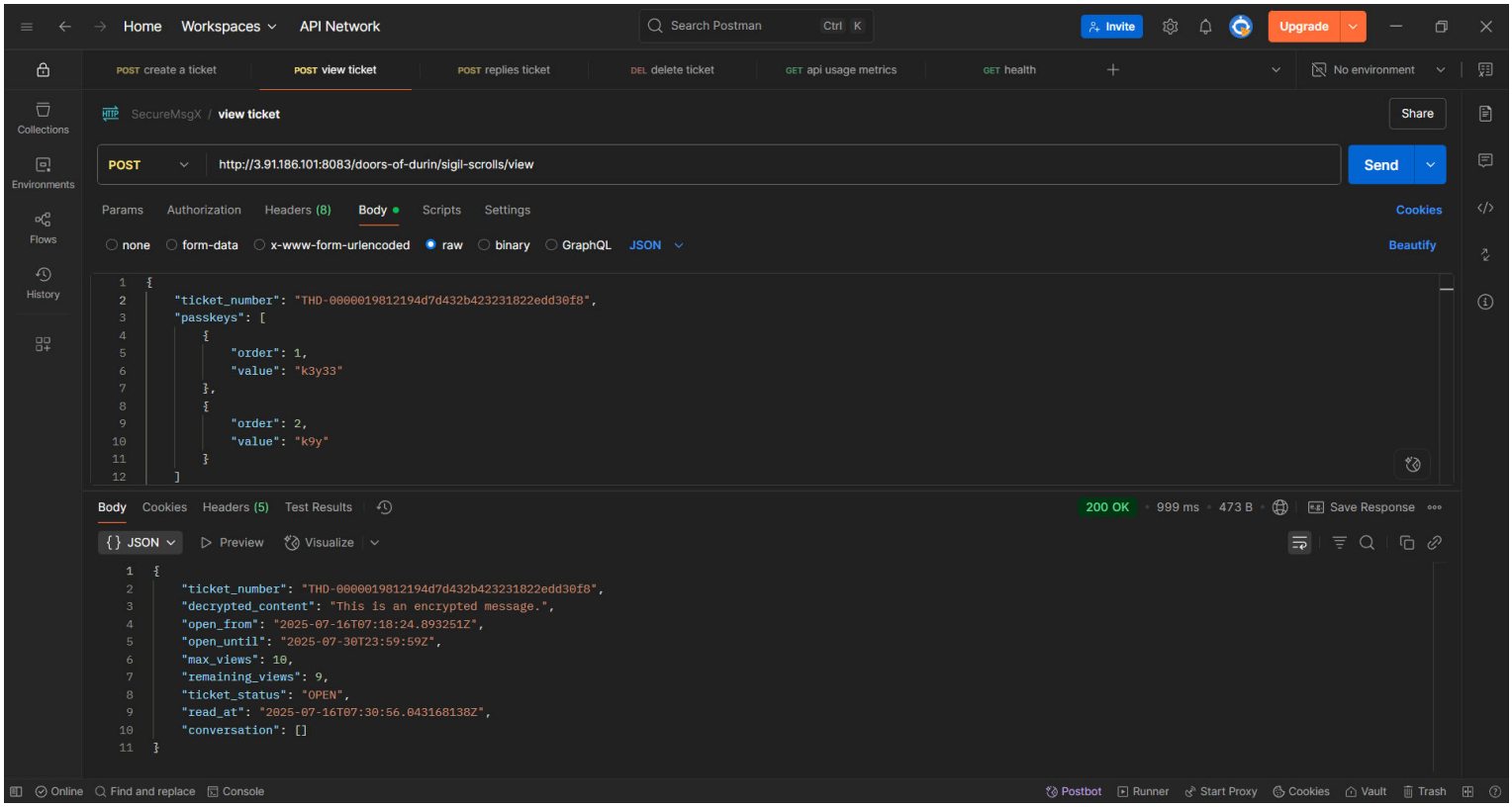
Payload:

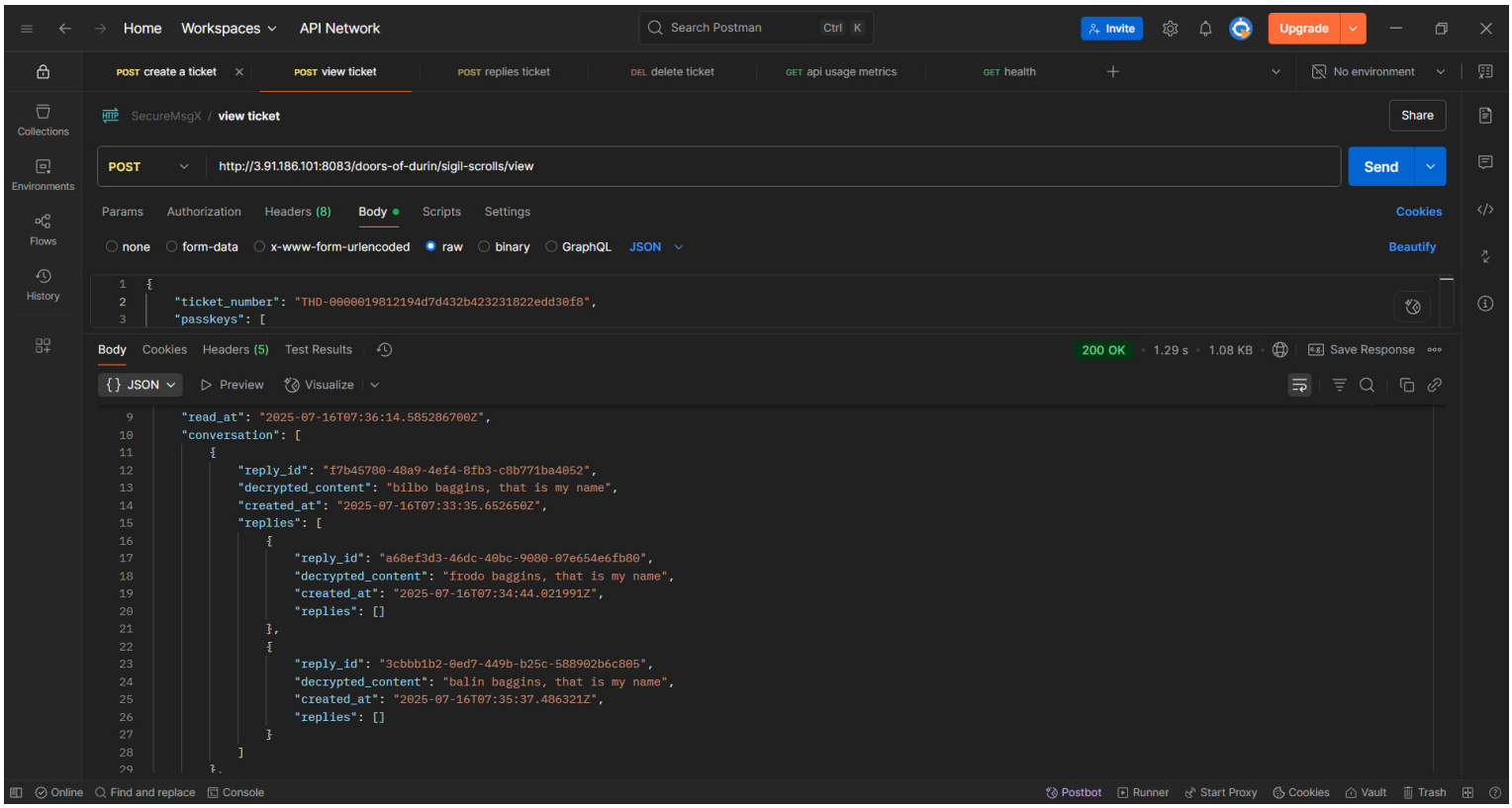
```
{
  "ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
  "passkeys": [
    {
      "order": 1,
      "value": "k3y33"
    },
    {
      "order": 2,
      "value": "k9y"
    }
  ]
}
```

Response:


```
{
   "ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
   "decrypted_content": "This is an encrypted message.",
   "open_from": "2025-07-16T07:18:24.893251Z",
   "open_until": "2025-07-30T23:59:59Z",
   "max_views": 10,
   "remaining_views": 8,
   "ticket_status": "OPEN",
   "read_at": "2025-07-16T07:36:14.585286700Z",
   "conversation": [
      {
         "reply_id": "f7b45780-48a9-4ef4-8fb3-c8b771ba4052",
         "decrypted_content": "bilbo baggins, that is my name",
         "created_at": "2025-07-16T07:33:35.652650Z",
         "replies": [
            {
               "reply_id": "a68ef3d3-46dc-40bc-9080-07e654e6fb80",
               "decrypted_content": "frodo baggins, that is my name",
               "created_at": "2025-07-16T07:34:44.021991Z",
               "replies": []
            },
            {
               "reply_id": "3cbbb1b2-0ed7-449b-b25c-588902b6c805",
               "decrypted_content": "balin baggins, that is my name",
               "created_at": "2025-07-16T07:35:37.486321Z",
               "replies": []
            }
         ]
      },
      {
         "reply_id": "6538c58e-deb2-4748-98a5-5031dd316e0b",
         "decrypted_content": "duan, that is my name",
         "created_at": "2025-07-16T07:36:08.112953Z",
         "replies": []
      }
   ]
}
```


📝 **Note:** All scrolls (tickets) can be viewed securely using the correct passkeys, but access is strictly governed by the view limits and access window (`open_from` to `open_until`).


Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
POST create a ticket
POST view ticket
POST replies ticket
DEL delete ticket
GET api usage metrics
GET health
No environment
Collections
Environments
Flows
History
SecureMsgX / view ticket
Share
POST
http://3.91.186.101:8083/doors-of-durin/sigil-scrolls/view
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
"passkeys": [
{
"order": 1,
"value": "k3y33"
},
{
"order": 2,
"value": "k9y"
}
]
Body
Cookies
Headers (5)
Test Results
200 OK
999 ms
473 B
Save Response
{} JSON
Preview
Visualize
{
"ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
"decrypted_content": "This is an encrypted message.",
"open_from": "2025-07-16T07:18:24.893251Z",
"open_until": "2025-07-30T23:59:59Z",
"max_views": 10,
"remaining_views": 9,
"ticket_status": "OPEN",
"read_at": "2025-07-16T07:30:56.043168138Z",
"conversation": []
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
POST create a ticket
POST view ticket
POST replies ticket
DEL delete ticket
GET api usage metrics
GET health
No environment
Collections
Environments
Flows
History
SecureMsgX / view ticket
Share
POST
http://3.91.186.101:8083/doors-of-durin/sigil-scrolls/view
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
"passkeys": [
Body
Cookies
Headers (5)
Test Results
200 OK
1.29 s
1.08 KB
Save Response
{} JSON
Preview
Visualize
"read_at": "2025-07-16T07:36:14.585286700Z",
"conversation": [
{
"reply_id": "f7b45780-48a9-4ef4-8fb3-c8b771ba4052",
"decrypted_content": "bilbo baggins, that is my name",
"created_at": "2025-07-16T07:33:35.652650Z",
"replies": [
{
"reply_id": "a68ef3d3-46dc-40bc-9080-07e654e6fb80",
"decrypted_content": "frodo baggins, that is my name",
"created_at": "2025-07-16T07:34:44.021991Z",
"replies": []
},
{
"reply_id": "3cbbb1b2-0ed7-449b-b25c-588902b6c805",
"decrypted_content": "balin baggins, that is my name",
"created_at": "2025-07-16T07:35:37.486321Z",
"replies": []
}
]
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash

💬 **How to reply (if allowed) (via /replies):** SecureMsgX allows users to reply to scrolls (tickets) that support conversations, such as `THREAD` and `GROUP`. You can post top-level replies directly to the scroll or nested replies in response to someone else's reply.

Endpoint: POST http://3.91.186.101:8083/doors-of-durin/sigil-scrolls/replies

Header: Content-Type: application/json

🧵 **Posting a Top-Level Reply:** You do not need to provide `parent_reply_id` when replying directly to the scroll.

Request:

```
{
  "ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
  "content": "duan, that is my name",
  "passkeys": [
    {
      "order": 1,
      "value": "k3y33"
    },
    {
      "order": 2,
      "value": "k9y"
    }
  ]
}
```

Response:

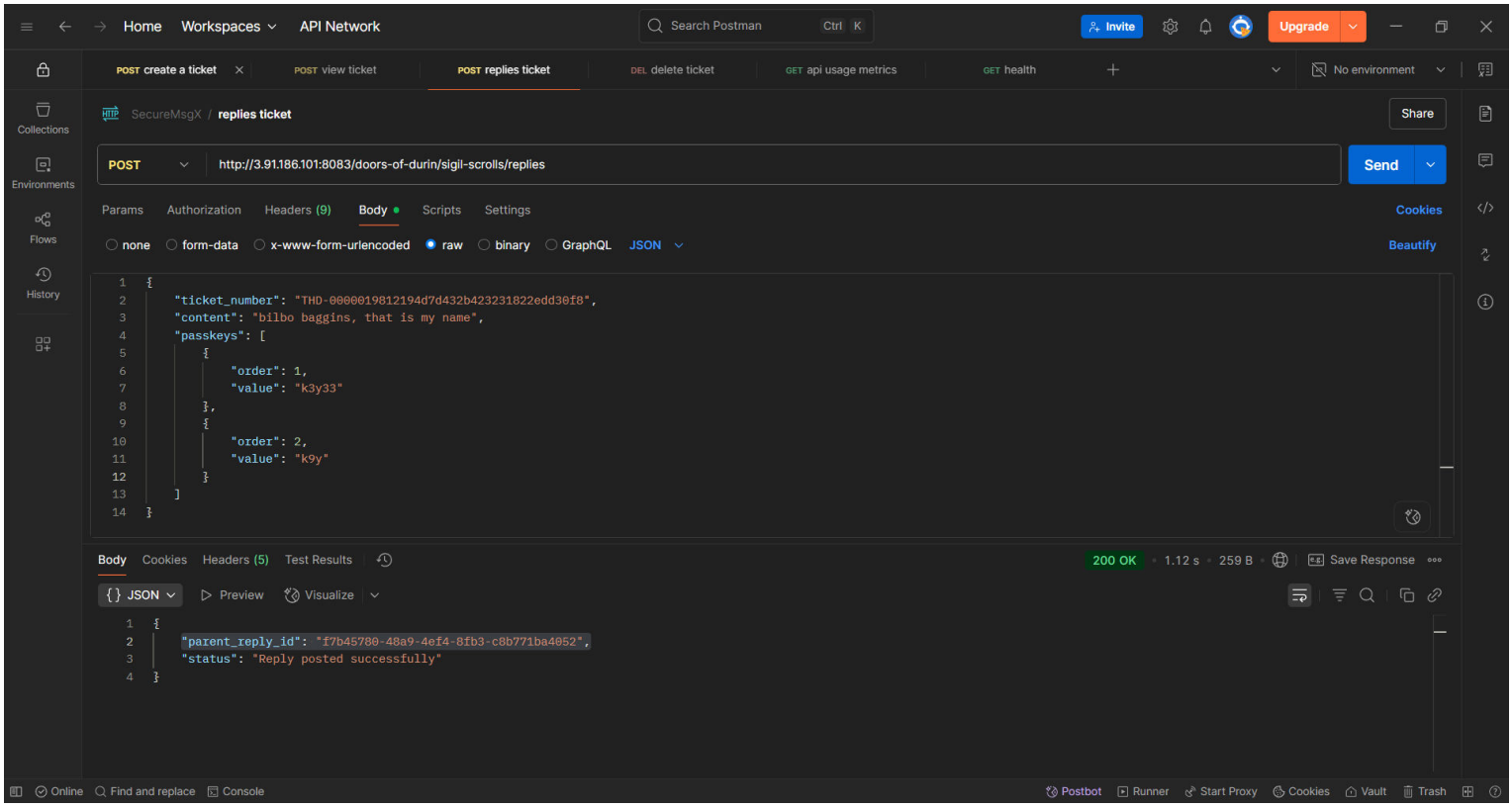
```
{
  "parent_reply_id": "6538c58e-deb2-4748-98a5-5031dd316e0b",
  "status": "Reply posted successfully"
}
```

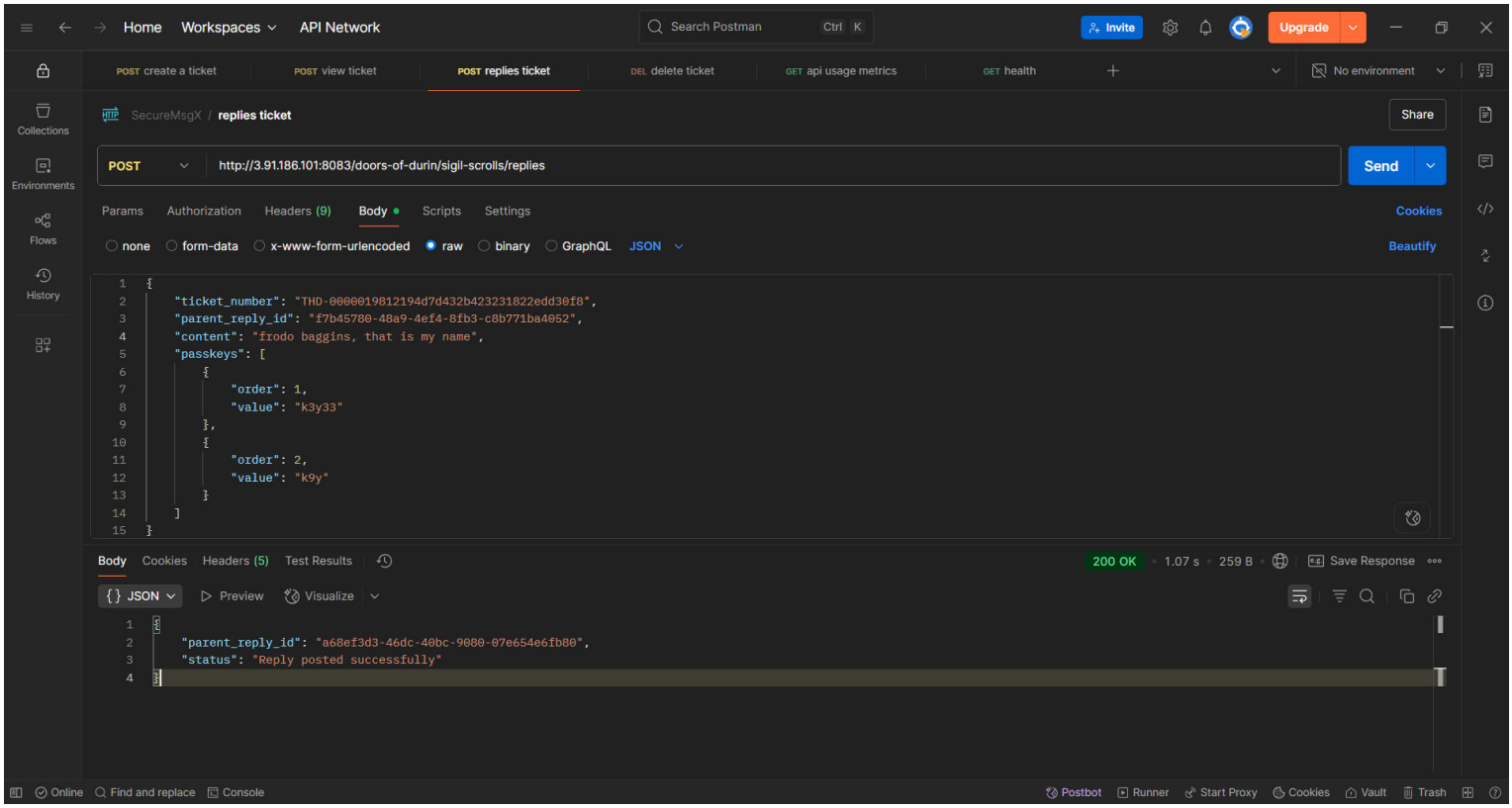
🔁 **Posting a Nested Reply:** To respond to a specific existing reply, include the `parent_reply_id` of the message you're replying to.

```
{
  "ticket_number": "THD-0000019812194d7d432b423231822edd30f8",
  "parent_reply_id": "f7b45780-48a9-4ef4-8fb3-c8b771ba4052",
  "content": " baggins, that is my name",
  "passkeys": [
    {
      "order": 1,
      "value": "k3y33"
    },
    {
      "order": 2,
      "value": "k9y"
    }
  ]
}
```

📝 Note:

- `THREAD` and `GROUP` ticket types allow replies.
- Use `parent_reply_id` **only when replying to another reply**, enabling nested conversation threads.



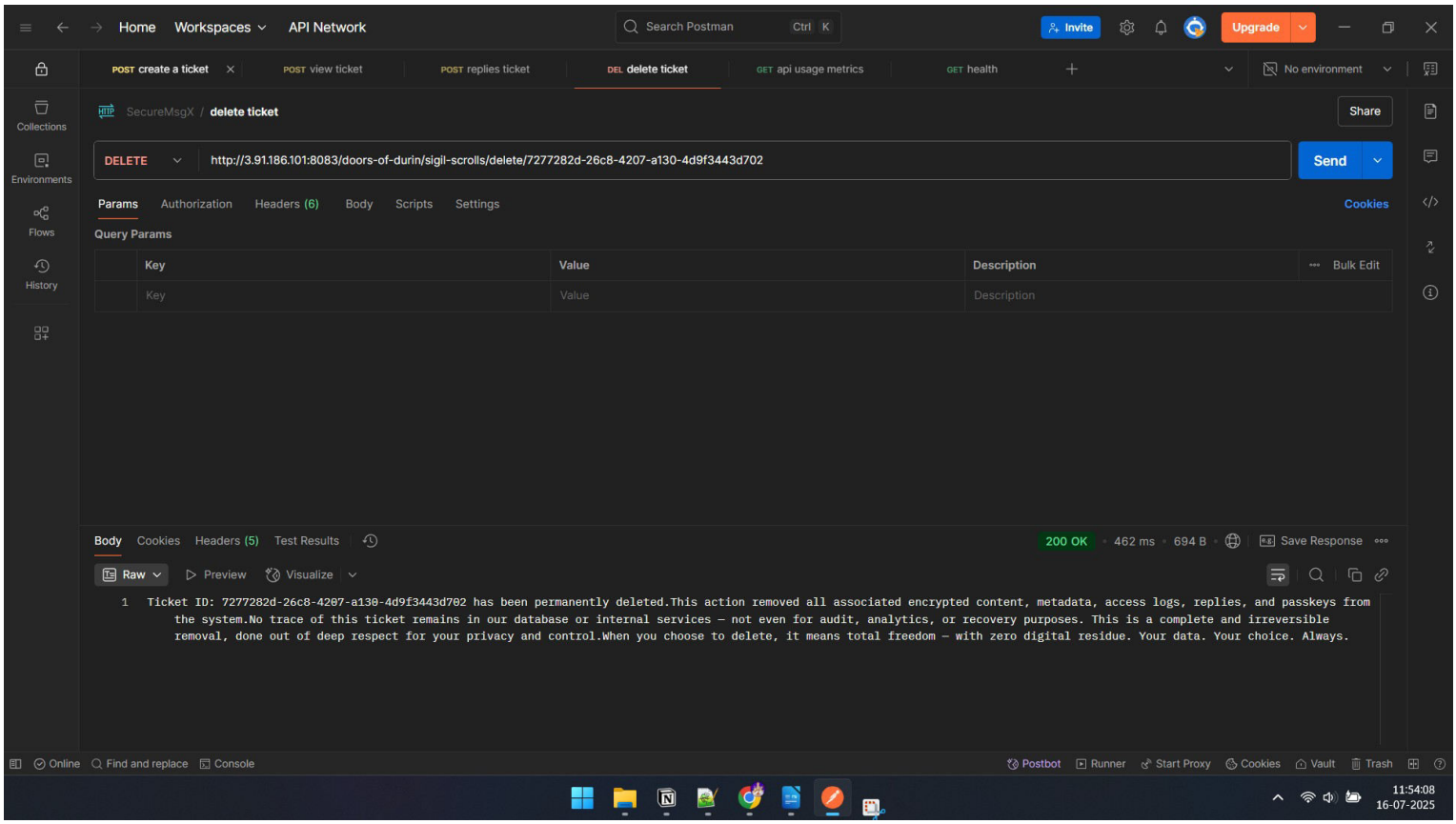


🗑️ **How to delete it (via /delete/{ticketId} — sender only):** SecureMsgX allows the original sender of a scroll to permanently delete it using its internal ticket_id. This operation is irreversible and ensures complete digital erasure of the scroll and all its related data.

Endpoint: DELETE http://3.91.186.101:8083/doors-of-durin/sigil-scrolls/delete/7277282d-26c8-4207-a130-4d9f3443d702

Request:

> Ticket ID: 7277282d-26c8-4207-a130-4d9f3443d702 has been permanently deleted. This action removed all associated encrypted content, metadata, access logs, replies, and passkeys from the system. No trace of this ticket remains in our database or internal services — not even for audit, analytics, or recovery purposes. This is a complete and irreversible removal, done out of deep respect for your privacy and control. When you choose to delete, it means total freedom — with zero digital residue. Your data. Your choice. Always.



📝 **Note:** When entering passkeys to view or reply to a scroll, **both the passkey values and their sequence order must be correct**. Even if all passkey values are correct, entering them in the wrong order will result in failure to decrypt the scroll. This is by design — to enhance security and prevent unauthorized access through guesswork or reordering.

> 🔐 **Security Reminder:**
> Your **passkey values** and their corresponding **order** (e.g., 1st, 2nd, etc.) together form the unique encryption key.
> ➤ *Both must match exactly what was used during ticket creation.*