Cross-Device Messaging API

|  |  |
| --- | --- |
| **HTTP Status Code** | **Meaning** |
| 200 | OK |
| 204 | OK – no contents to return |
| 4000 | Bad credentials |
| 4001 | Asset exists |
| 4002 | Invalid entry |

|  |  |
| --- | --- |
| **JSON Error Code** | **Meaning** |
| DNE | Does not exist |
| UAE | User already exists |
| IVU | Invalid username |
| PTS | Password too short |
| PTL | Password too long |
| IVP | Invalid password |
| NNP | No new password |
| INP | Incorrect password/credentials |

This RESTful API contains six methods:

* /usermod
  + /register
  + /check
  + /passwd
  + /delete
* /messaging
  + /send
  + /fetch

The methods under usermod allow for creating a new user, testing user credentials, changing passwords, and deleting users, respectively. The methods under messaging allow for sending and fetching messages, respectively. All methods use HTTP POST only, allowing for a more secure channel of communication when used in conjunction with a HTTP server capable of HTTPS.

This RESTful API accepts and provides JSON in HTTP requests. Each method is documented in the following pages.

**/usermod/register**

Function: allow for the registration of new users

This method accepts data in JSON format. In order, the data should include:

id: The username of the new user

passwd: The password of the new user

User accounts must have usernames between 4-16 characters long. Passwords must be between 8-32 characters long. Both must be alphanumeric, with the exception of ‘-’ and ‘\_’ characters. An HTTP status of 200 will be returned upon the successful creation of an account. Otherwise, an error code in JSON format and an alternate HTTP status will be returned. Refer to page 1 to resolve these codes.

**/usermod/check**

Function: allow for testing user credentials

This method accepts data in JSON format. In order, the data should include:

id: The username to be tested

passwd: The password to be tested

If the credentials are correct, an HTTP status of 200 will be returned with no further data. Otherwise, an HTTP status of 4000 will be returned with no data. This method is useful for testing new applications to interface with the API.

**/usermod/passwd**

Function: allow for the changing of passwords of existing users

This method accepts data in JSON format. In order, the data should include:

id: The username of the user to have its password changed

passwd: The old password

newPasswd: The new password

Passwords must be between 8-32 characters long. They must be alphanumeric, with the exception of ‘-’ and ‘\_’ characters. Refer to page 1 to resolve any error or status code.

**/usermod/delete**

Function: allow for the deletion of existing users

This method accepts data in JSON format. In order, the data should include:

id: The username of the user to be deleted

passwd: The password of the user to be deleted

Accounts are deleted immediately after this request. However, conversation data is not deleted.

**/messaging/send**

Function: allow for the sending of messages to other users

This method accepts data in JSON format. In order, the data should include:

sender: The username of the user sending the message

senderPass: The password of the user sending the message

receiver: The username of the recipient of the message

msgBody: The body of the message to be sent to the receiver

Messages cannot be sent to invalid recipients.

**/messaging/fetch**

Function: allow for the retrieval of messages

This method accepts data in JSON format. In order, the data should include:

id: The username of the user retrieving the messages

passwd: The password of the user retrieving the messages

The last 100 messages from each conversation are contained in the response if the authentication is successful. Messages in the response are in reverse order.