**Nexus Chat Audit API Documentation**

1. **Database Structure**
   1. Audit Table
      1. This is table is used to store information regarding sessions, user, workgroup, location and reasons.
      2. Table schema as follows

|  |
| --- |
| Audit |
| id (PK) |
| dept\_name |
| loc\_name |
| reason\_name |
| session\_id |
| session\_timestamp |
| status |
| user\_name |
| user\_uid |
| session\_attributes |
| session\_closure |
| user\_email |

* + 1. SQL Query

**CREATE** **TABLE** `audit` (

`id` **INT**(11) **NOT** **NULL** **AUTO\_INCREMENT**,

`dept\_name` **VARCHAR**(255) **NOT** **NULL**,

`loc\_name` **VARCHAR**(255) **NOT** **NULL**,

`reason\_name` **VARCHAR**(255) **NOT** **NULL**,

`session\_id` **INT**(11) **NOT** **NULL**,

`session\_timestamp` **DATETIME** **NOT** **NULL**,

`status` **VARCHAR**(15) **NOT** **NULL**,

`user\_name` **VARCHAR**(255) **NOT** **NULL**,

`user\_uid` **VARCHAR**(255) **NOT** **NULL**,

`session\_attributes` **VARCHAR**(255) **NOT** **NULL**,

`session\_closure` **VARCHAR**(255) **NOT** **NULL**,

`user\_email` **VARCHAR**(100) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

)

* 1. Status\_Audit Table
     1. This table is used to store chat related statuses, time of status with respective user
     2. Table schema as follows

|  |
| --- |
| Status\_audit |
| id (PK) |
| status |
| status\_time |
| user\_uid |

* + 1. SQL Query

**CREATE** **TABLE** `status\_audit` (

`id` **INT**(11) **NOT** **NULL** **AUTO\_INCREMENT**,

`status` **VARCHAR**(15) **NOT** **NULL**,

`status\_time` **DATETIME** **NOT** **NULL**,

`user\_uid` **VARCHAR**(255) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

)

1. **API Details**
   1. To save audits of session, departments and users etc.
      1. End point : /audit/save
      2. Request Payload:

{

"status":"Active",

"sessionId":545,

"userUid":"AB12345",

"reasonName":"BSW",

"deptName":"Service Assurance CRIS Prism",

"locName":"NULL",

"userName":"Test",

"userEmail":"test\_user@straviso.com",

"sessionClosure":"sessionClosure",

"sessionAttributes":"sessionAttributes"

}

* + 1. Response:
       1. Success:

{

"message": "Success",

"data": {

"id": 0,

"status":"Active",

"sessionId":545,

"userUid":"AB12345",

"reasonName":"BSW",

"deptName":"Service Assurance CRIS Prism",

"locName":"NULL",

"userName":"Test",

"userEmail":"test\_user@straviso.com",

"sessionClosure":"sessionClosure",

"sessionAttributes":"sessionAttributes"

},

"code": 200

}

* + - 1. Error:

{

"message": "<respective failure message>",

"data": null,

"code": 200

}

* 1. To save status audits related to chats
     1. End Poing: /audit/status/save
     2. Request Payload:

{

"status":"Active",

"user\_uid":"125"

}

* + 1. Response:
       1. Success:

{

"message": "Success",

"data": {

"id":"0",

"user\_uid":"125",

"status":"Active",

"statusTime": 1500296828336

},

"code": 200

}

* + - 1. Error

{

"message": "<respective failure message>",

"data": null,

"code": 200

}

1. **Execution of API**
   1. To save audit records request first comes to method controller with payload data which will store into database table.
   2. Then it passes payload data to service and from service to DAO layer to store into the database.
   3. When audit data store it will take time to save data into the database as audit; chat functionality will wait until it store audit.
   4. So to avoid this we are using threading mechanism
      1. Using threading we give data to thread and thread will execute and store data into database table
      2. So due to this it will work as asynchronous and when chatting is ongoing so it will not wait to store audit data and send chat messages.
      3. Whole process will work as asynchronously.