Introduce GenreX API

A Generative Music AI for Your Application

Welcome to GenreX API for generative AI music! 🎵

At GenreX, we are passionate about blending technology with creativity to transform music. This API service is crafted for developers who seek to integrate state-of-the-art generative AI music into their applications, enhancing user experiences with unique and high-quality soundscapes. 🚀

Designed for a broad range of applications—from gaming to video creation—GenreX API uses advanced AI algorithms to compose and generate adaptive music in real-time. Whether you are looking to enrich your app's audio environment or create personalized sound experiences, GenreX API is here to help you achieve your vision.

Happy building! 🛠️

To apply for API access, please book a meeting here.

For feedback or inquiries, please email us at help@genrex.com.

To explore our offerings further, please visit: GenreX App.

How To Get an API Key?

We would love to learn more about what you are building and how we could reshape music composition together. If you are interested in intergrating with GenreX, book a call with us today! Our team is ready to assist you in seamlessly integrating our innovative music solutions into your projects. 🎶📞

👉 Book meeting here.

Quick Start

PREREQUISITE

To access GenreX API, you must obtain the keys from the platform. Please contact us. to apply for an API key.

To generate music via API, follow these two steps:

1. Request the generation and immediately receive the `queryID` via the endpoint `v1/text2music/generateMusic`.
2. Obtain the audio URL by continuously checking the query status using the `queryID` in a loop until it is completed via the endpoint `v1/text2music/retrieve`.

Both steps require a user signature to verify identity.

Here is a sample code snippet to get you started

Request Generation

Construct the payload

Suppose you want to create a 10-second song by prompting something like "intense EDM," your API request payload body should be like this:

Node.js Python

```
const payload = JSON.stringify({
  duration: 10,
  text: "intense EDM",
});
```

Create signature

To avoid directly passing the API secret to our API and prevent forgery, you can create the signature by encrypting `${timestamp}.${payload}` with the API secret using the SHA-256 HMAC algorithm.

Node.js Python

```
const timestamp = Date.now();
const dataToSign = `${timestamp}.${payload}`;
const hmac = crypto.createHmac("sha256", apiSecret);
const signature = hmac.update(dataToSign).digest("hex");
```

Create headers

Node.js Python

```
const headers = {
  "gx-key": apiKey,
  "gx-signature": `t=${timestamp},v=${signature}`,
  "Content-Type": "application/json",
};
```

Send request and receive queryId

Node.js Python

```
axios
  .post("https://api.genrex.com/v1/text2music/generateMusic", payload, {
    headers,
  })
  .then((response) => {
      console.log("queryId:", response.data.id);
  })
  .catch((error) => {
    console.error(
      "Error:",
      error.response ? error.response.data : error.message
    );
  });
```

Let's put everything together 😃✌️

Node.js   Python

```javascript
const crypto = require("crypto");
const axios = require("axios");
const apiKey = "<YOUR-API-KEY>";
const apiSecret = "<YOUR-API-SECRET>";

// construct the payload
const payload = JSON.stringify({
  duration: 10,
  text: "intense EDM",
});

// create signature
const timestamp = Date.now();
const dataToSign = `${timestamp}.${payload}`;
const hmac = crypto.createHmac("sha256", apiSecret);
const signature = hmac.update(dataToSign).digest("hex");

// create headers
const headers = {
  "gx-key": apiKey,
  "gx-signature": `t=${timestamp},v=${signature}`,
  "Content-Type": "application/json",
};

// Send post request
axios
  .post("https://api.genrex.com/v1/text2music/generateMusic", payload, {
    headers,
  })
  .then((response) => {
    console.log("queryId:", response.data.id);
  })
  .catch((error) => {
    console.error(
      "Error:",
      error.response ? error.response.data : error.message
    );
  });
```

Obtain the audio assets

Once you submit the generation request, you can use the `queryID` to check if the generation is ready. Similarly, we request signature as logic in request generation.

Construct the payload

Node.js Python

```
const createPayload = (queryId) => JSON.stringify({queryId});
```

Create signature

Node.js Python

```
const createSignature = (apiSecret, dataToSign) => {
  const hmac = crypto.createHmac("sha256", apiSecret);
  return hmac.update(dataToSign).digest("hex");
};
```

Create headers

Node.js Python

```
const createHeaders = (apiKey, apiSecret, payload) => {
  const timestamp = Date.now();
  const dataToSign = `${timestamp}.${payload}`;
  const signature = createSignature(apiSecret, dataToSign);
  return {
    "gx-key": apiKey,
    "gx-signature": `t=${timestamp},v=${signature}`,
    "Content-Type": "application/json",
  };
};
```

Check the status of the generation

Node.js Python

```
const checkStatus = async (payload, headers) => {
  try {
    const response = await axios.post(
      "https://api.genrex.com/v1/text2music/retrieve",
      payload,
      { headers }
    );

    switch (response.data.status) {
      case "Completed":
        return response.data.response[0].content.url;
      case "Processing":
        console.log("Generation in progress, checking again...");
        return null;
      case "Failed":
        console.error(
          "The generation failed. Please submit a new generation request."
        );
        process.exit(1);
      default:
        console.log("Unknown status, exiting.");
        process.exit(1);
    }
  } catch (error) {
    console.error(
      "Error checking status:",
      error.response ? error.response.data : error.message
    );
    throw error;
  }
};
```

Loop the checking process

Node.js Python

```
// Main function to get the audio URL
const getAudioUrl = async (apiKey, apiSecret, queryId) => {
  const pollingInterval = 5000; // Set the polling interval in milliseconds
  const payload = createPayload(queryId);
  const headers = createHeaders(apiKey, apiSecret, payload);

  while (true) {
    const audioUrl = await checkStatus(payload, headers);
    if (audioUrl) return audioUrl;

    await new Promise((resolve) => setTimeout(resolve, pollingInterval));
  }
};
```

Let's put everything together AGAIN 😀✌️

Node.js Python

```
const axios = require("axios");
const crypto = require("crypto");

const apiKey = "<YOUR-API-KEY>";
const apiSecret = "<YOUR-API-SECRET>";
const queryId = "<YOUR-QUERY-ID>"; // Replace this with your actual Query ID

// Function to create payload with the queryId
const createPayload = (queryId) => JSON.stringify({ queryId });

// Function to create signature
const createSignature = (apiSecret, dataToSign) => {
  const hmac = crypto.createHmac("sha256", apiSecret);
  return hmac.update(dataToSign).digest("hex");
};

// Function to create headers
const createHeaders = (apiKey, apiSecret, payload) => {
  const timestamp = Date.now();
  const dataToSign = `${timestamp}.${payload}`;
  const signature = createSignature(apiSecret, dataToSign);
  return {
    "gx-key": apiKey,
    "gx-signature": `t=${timestamp},v=${signature}`,
    "Content-Type": "application/json",
  };
};
```

```javascript
// Function to check the status of the generation
const checkStatus = async (payload, headers) => {
  try {
    const response = await axios.post(
      "https://api.genrex.com/v1/text2music/retrieve",
      payload,
      { headers }
    );

    switch (response.data.status) {
      case "Completed":
        return response.data.response[0].content.url;
      case "Processing":
        console.log("Generation in progress, checking again...");
        return null;
      case "Failed":
        console.error(
          "The generation failed. Please submit a new generation request."
        );
        process.exit(1);
      default:
        console.log("Unknown status, exiting.");
        process.exit(1);
    }
  } catch (error) {
    console.error(
      "Error checking status:",
      error.response ? error.response.data : error.message
    );
    throw error;
  }
};

// Main function to get the audio URL
const getAudioUrl = async (apiKey, apiSecret, queryId) => {
  const pollingInterval = 5000; // Set the polling interval in milliseconds
  const payload = createPayload(queryId);
  const headers = createHeaders(apiKey, apiSecret, payload);

  while (true) {
    const audioUrl = await checkStatus(payload, headers);
    if (audioUrl) return audioUrl;

    await new Promise((resolve) => setTimeout(resolve, pollingInterval));
  }
};
```

```
// Self-invoking function to use getAudioUrl
(async () => {
  try {
    const audioUrl = await getAudioUrl(apiKey, apiSecret, queryId);
    console.log("Audio URL:", audioUrl);
  } catch (error) {
    console.error("Failed to retrieve audio URL:", error);
  }
})();
```

How to Check Your Balance?

We highly recommend that users frequently self-monitor their remaining balance. Here is the code to check your quota and credits:

Node.js Python

```
const crypto = require("crypto");
const axios = require("axios");
const apiKey = "<YOUR-API-KEY>"; // Replace this with your actual API Key
const apiSecret = "<YOUR-API-SECRET>"; // Replace this with your actual API
Secret

// create signature
const timestamp = Date.now();
const payload = JSON.stringify({});
const dataToSign = `${timestamp}.${payload}`;
const hmac = crypto.createHmac("sha256", apiSecret);
const signature = hmac.update(dataToSign).digest("hex");

// create headers
const headers = {
  "gx-key": apiKey,
  "gx-signature": `t=${timestamp},v=${signature}`,
  "Content-Type": "application/json",
};

// Send post request
axios
  .post("https://api.genrex.com/v1/text2music/getBalance", payload, {
    headers,
  })
```

```
    .then((response) => {
      console.log("Response:", response.data);
    })
    .catch((error) => {
      console.error(
        "Error:",
        error.response ? error.response.data : error.message
      );
    });
```

As a result, you will receive a response like this:

```
{
  "durationLimit": 60,
  "creditQuota": 54,
  "usedCredit": 0,
  "userId": "a6f434b6-a475-40b1-81e3-3ce9868945ff",
  "createdAt": "2024-06-01T16:47:17.054Z",
  "updatedAt": "2024-06-03T03:26:42.735Z"
}
```

Request Generation

Submit a generation request to the backend

POST `https://api.genrex.com/v1/text2music/generateMusic`

Request body

duration integer Required

- The duration of music. The value must be between 5 seconds and 60 seconds.

```
{
  "duration":10,
  "text":"music"
}
```

text string Required

- The prompt text of the generation. The string must be between 1 and 250 characters.

Headers

gx-key string Required

- your-api-key

```
{
  'gx-key': '4nBmZkU2-VdTNrPqL7e9y', //
7Xd9PfMlkanB5ZTuj0qQG
  'gx-signature':
't=1717574785354,v=c3eaaf9b064283cd3558
  'Content-Type': 'application/json' ,
}
```

gx-signature string Required

- To generate the `gx-signature`, follow these steps:
    i. Convert the payload object to a JSON string.
    ii. Concatenate the current timestamp and the JSON string with a dot (.) to form the string to be signed e.g. `${timestamp}.${payload}`
    iii. Use the `sha256-HMAC` algorithm and `your-api-secret` to sign the string, generating a hexadecimal string signature.
    iv. The `gx-signature` field value should be formatted as `t=timestamp,v=signature`.

Content-Type string Required

- Must be set to application/json.

Responses

- response.data

id string

- The id is used to check if the generation query is complete. Recommend save locally.

processSystem string

- The versioning of AI

source string

- The source of the request

```
{
  id: 'QRY_WRXzIXyidD8x9xIEZZ4GppXh',
  processSystem: 'GenrexV1',
  source: 'API',
  status: 'Processing',
  contentType: 'Text',
  content: { text: 'music', duration:
30 },
  metadata: null,
  userId: 'a6f434b6-a475-40b1-81e3-
3ce9868945ff',
  createdAt: '2024-06-
02T16:16:07.386Z',
```

status string

- The status of the generation, which can be either `Processing`, `Failed`, or `Completed`.

```
  updatedAt: '2024-06-
02T16:16:07.386Z'
}
```

content object

metadata array or null

userId string

createdAt string

- The timestamp when the query was created

updatedAt string

- The timestamp when the status was last updated

Audio Retrival

Retrieve audio assets and generation status by `queryid`

POST `https://api.genrex.com/v1/text2music/retrieve`

Request body

queryId string Required

- This queryId is obtained from the initial generation request and is used to track the status and retrieve the generated audio content.

```
{
"queryId":"QRY_WRXzIXyidD8x9xIEZZ4GppXh
}
```

Headers

gx-key string Required

- your-api-key

gx-signature string Required

- To generate the `gx-signature`, follow these steps:

  i. Convert the payload object to a JSON string.
  ii. Concatenate the current timestamp and the JSON string with a dot (.) to form the string to be signed e.g. `${timestamp}.${payload}`
  iii. Use the `sha256-HMAC` algorithm and `your-api-secret` to sign the string, generating a hexadecimal string signature.
  iv. The `gx-signature` field value should be formatted as `t=timestamp,v=signature`.

Content-Type string Required

- Must be set to application/json.

```
{
  "gx-key": "4nBmZkU2-VdTNrPqL7e9y", //
7Xd9PfMlkanB5ZTuj0qQG
  "gx-signature":
"t=1717576160738,v=8142c06454544f0cfd77
  "Content-Type": "application/json"
}
```

Responses

- response.data

id string

- underlying `queryId`

processSystem string

source string

status string

- The status of the generation, which can be either `Processing`, `Failed`, or `Completed`. Only when the status is `Completed` will the object return the audio assets.

content object

```
{
  "id": "QRY_WRXzIXyidD8x9xIEZZ4GppXh",
  "processSystem": "GenrexV1",
  "source": "API",
  "status": "Completed",
  "contentType": "Text",
  "content": {
    "text": "music",
    "duration": 30
  },
  "metadata": null,
  "userId": "a6f434b6-a475-40b1-81e3-
3ce9868945ff",
  "createdAt": "2024-06-02T11:04:19.986
  "updatedAt": "2024-06-02T11:05:21.826
  "response": [
    {
      "id": "RSP_Q4YGrD4wq0F8U40AbioCui
```

metadata array or null

userId string

createdAt string

updatedAt string

response array

- The main array for audio assets retrieval.

id string

- The `responseId`

metadata array

volume_visualize array

- The normalized volume by timestep. The scale is from 0 to 127. There are 1000 discrete segments to represent the continuous audio object.

contentType string

content array

url string

- The URL of the generated audio file.

queryId string

createdAt string

updatedAt string

```
      "metadata": [
        {
          "volume_visualize": [
            67, 76, 77, 73, 84, 87, 83,
94, 89, 85,
            109, 92, 77, 69, 91, 85, 81
88, 88, 93,
            107, 99, 97, 88, 95, 82, 68
104, 111, 86, 95,
            97, 88, 85, 95, 97, 110, 10
93, 105, 125, 107,
            96, 104, 106, 103, 103, 108
122, 89, 102, 101, 101,
            103, 101, 122, 117, 96, 96,
105, 94, 105, 103, 111,
            97, 96, 91, 100, 86, 100, 1
104, 109, 112, 111,
            93, 70, 91, 108, 110, 104,
104, 93, 86, 93,
            94, 103, 98, 96,
            // ... 900 more items
          ]
        }
      ],
      "contentType": "Audio",
      "content": [
        {
          "url":
"https://octaimusic.s3.amazonaws.com/te
test/1717326321.3117287_32k.wav"
        }
      ],
      "queryId": "QRY_WRXzIXyidD8x9xIEZ
      "createdAt": "2024-06-02T11:05:21
      "updatedAt": "2024-06-02T11:05:21
    }
  ]
}
```

Get Balance

Retrieve the account generation credit balance

POST `https://api.genrex.com/v1/text2music/getBalance`

Request body

> The submitted `${payload}` must be assigned as a string in the form of `{}`

```
{}
```

Headers

> **gx-key** string Required
>
> - your-api-key
>
> **gx-signature** string Required
>
> - To generate the `gx-signature`, follow these steps:
>
>   i. Convert the payload object to a JSON string.
>   ii. Concatenate the current timestamp and the JSON string with a dot (.) to form the string to be signed e.g. `${timestamp}.${payload}`
>   iii. Use the `sha256-HMAC` algorithm and `your-api-secret` to sign the string, generating a hexadecimal string signature.
>   iv. The `gx-signature` field value should be formatted as `t=timestamp,v=signature`.
>
> **Content-Type** string Required
>
> - Must be set to application/json.

```
{
  'gx-key': '4nBmZkU2-VdTNrPqL7e9y', //
7Xd9PfMlkanB5ZTuj0qQG
  'gx-signature':
't=1717590783542,v=503eb068b5cddc2f3a23
  'Content-Type': 'application/json' ,
}
```

Responses

- response.data

> **maxDurationSeconds** integer

- The maximum duration (in seconds) allowed for each API session.

totalCreditQuota integer

- The total amount of credits allocated to the user. This quota represents the maximum credits that can be used by the user.

creditsUsed integer

- The amount of credits that have already been used by the user.

userId string

createdAt string

updatedAt string

```
{
  maxDurationSeconds: 60,
  totalCreditQuota: 95,
  creditsUsed: 0,
  userId: 'a6f434b6-a475-40b1-81e3-
3ce9868945ff',
  createdAt: '2024-06-
01T16:47:17.054Z',
  updatedAt: '2024-06-
08T11:46:15.594Z'
}
```