



SGMETADATA API DOCUMENTATION

REV 0.0.3 - 12/12/2016

TABLE OF CONTENTS

Introduction.....	1
Integration Requirements	1
REST API JavaScript Client Integration Procedure	2
REST API JavaScript Client.....	3
Constructor: SGMetadata(config)	3
Function: SGMetadata.getCurrent(uuid, callback(response){})	4
Function: SGMetadata.getMetadata(uuid, callback(responseArray){}).....	5
Function: SGMetadata.getCurrentRaw(UUID, callback(response){}).....	6
Function: SGMetadata.getMetadataRaw(UUID, callback(responseArray){}).....	7
REST API Endpoints.....	8
Endpoint: Current Metadata	8
Endpoint: All Metadata	9
Endpoint: Current Raw Metadata	10
Endpoint: All Raw Metadata	10
Socket Integration Procedure.....	11
Socket.io Client Integration	12
SGmetadata Socket API	12
Socket.io Functions.....	12
Function: io.connect()	12
Function: socket.disconnect()	13

Socket Emit Channels	13
Channel: listen	13
Channel: stop.....	14
Channel: getMetadata.....	14
Channel: getScraperInfo.....	15
Socket Events.....	15
Event: connect.....	15
Event: metadata	16
Event: rawMetadata.....	16
Event: scraperInfo	17
Event: disconnect	17
SGmetadata Socket Integration Example	18

INTRODUCTION

This document covers client-side JavaScript integration for the SGmetadata service using REST API requests and Socket.io web sockets. After performing the integration steps outlined in this document, your page will display in-stream metadata by a) Requesting metadata at load time, or b) Opening a persistent socket connection to SGmetadata at load time and handling metadata update push events from the server.

INTEGRATION REQUIREMENTS

In order to integrate with SGmetadata you will need the following:

- 1) An account on SGmetadata with at least 1 metadata scraper running.
- 2) A SGmetadata API key. This will be provided to you by StreamGuys technical support. This key will grant your page(s) access to the SGmetadata API.

- 3) One SGmetadata scraper UUID per station you wish to retrieve metadata for.
These will be provided to you by StreamGuys technical support. These ids allow you to request metadata for a specific scraper in the system.
- 4) If performing a push-based integration, the Socket.io JavaScript client must be included in the head of your page. See the [Socket.io Client Integration](#) section for more details.

REST API JAVASCRIPT CLIENT INTEGRATION PROCEDURE

If you wish to write your own custom REST API client, skip to the [REST API Endpoints](#) section. Otherwise, perform the following steps to add SGmetadata REST API support to your page.

- 1) Download the SGMetadata client JavaScript file from:
<https://jet.streamguys.com/js/sgmetadata-client.js>
- 2) Include the SGMetadata client JavaScript in the head of your page:

```
<script src="sgmetadata-client.js" type="text/javascript"></script>
```

- 3) Configure and invoke the SGMetadata JavaScript client using your scraper UUID and account API key:

```
<script type="text/javascript">
  //This is the UUID for the example scraper.
  var scraperUUID = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx';
  //This is your SGmetadata account API key.
  var accountApiKey = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';

  //Build an API client instance with your specific api key.
  //This client abstracts the HTTP endpoints for the SGmetadata api.
  var sgmd = SGMetadata({apiKey: accountApiKey});

  //At this point, you can use the sgmd instance to make API requests.
</script>
```

- 4) Perform your API request(s) for metadata and handle the response:

```
<script type="text/javascript">
  //This is the UUID for the example scraper.
  var scraperUUID = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx';
  //This is your SGmetadata account API key.
  var accountApiKey = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';

  //Build an API client instance with your specific api key.
```

```
//This client abstracts the HTTP endpoints for the SGmetadata api.
var sgmd = SGMetadata({apiKey: accountApiKey});

//Fetch an array of all metadata objects for the metadata scraper.
sgmd.getMetadata(scraperUUID, function(metadataArray) {
    //Handle the array of metadata objects from the api response.
    yourMetadataDisplayFunction(metadataArray);
});
</script>
```

- 5) Read through the [REST API JavaScript Client](#) section for additional descriptions of the functions available to the SGMetadata client.

REST API JAVASCRIPT CLIENT

The functions available to the JavaScript SGMetadata client are detailed below. The client may be downloaded here: <https://jet.streamguys.com/js/sgmetadata-client.js>

A live example may be seen here: <http://example.streamguys.com/sgmetadata/api.html>

Constructor: SGMetadata(config)

Invoke the SGMetadata constructor to create a new SGMetadata API client instance that can be used to issue API requests.

PARAMETERS:

- 1) **(Object) config** – The main SGMetadata client configuration object. This object extends and overrides the default settings of the SGMetadata class. The config object may have the following properties:
 - a. **[Required] (String) apiKey** – The SGMetadata account API key to use for this instance's API requests. This config parameter is required for any API request functions to succeed.
 - b. **[Optional] (String) apiBase** – The base URL string to use when constructing api request endpoints. Includes protocol, domain, and port. Only change this if instructed to by StreamGuys. Example: `https://jeta.streamguys.com:8444`
 - c. **[Optional] (Object) endpoints** – An object that maps endpoint names to REST API endpoint path strings. Used to construct the final portion of the api

request URL string. Only change this if instructed to by StreamGuys.

Example:

```
endpoints: {  
  //Returns a single metadadata object  
  current: '/metadata',  
  //Returns the entire metadata array of a scraper  
  metadata: '/metadata/all',  
  //Returns a single raw metadata string  
  currentRaw: '/metadata/raw',  
  //Returns the entire array of raw metadata strings.  
  metadataRaw: '/metadata/raw/all'  
}
```

RESULTS:

The constructor returns a new SGMetadata class instance that has access to the rest of the functions detailed in this section.

EXAMPLE:

```
//Paste your SGmetadata API key here.  
var accountApiKey = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
//Build an API client instance with your specific api key.  
//This client instance can be used to invoke API request functions.  
var sgmd = SGMetadata({apiKey: accountApiKey});
```

Function: SGMetadata.getCurrent(uuid, callback(response){})

This function requests the most recent metadata object from the scraper matching the given UUID.

PARAMETERS:

- 1) **(String) uuid** – The SGmetadata unique id string matching the SGmetadata scraper you wish to request metadata for.
- 2) **(Function) callback(response)** – The function to invoke and pass a response object to when the API request is complete.

RESULTS:

This function's callback is passed a response object that has the following properties:



- EXAMPLE:

Function: SGMetadata.getMetadata(uuid, callback(responseArray){})

[5]

RESULTS:

The **responseArray** passed to the callback is an array of metadata strings the same as the single string returned by the **getCurrentRaw** function. By default, the **responseArray** is in chronological order with the most recent (now playing) metadata string at index 0, and the previously played song metadata strings at indices 1 to n where n is the total number of metadata items tracked by the scraper matching the given UUID.

REST API ENDPOINTS

The SGmetadata REST API endpoints are detailed below. For each endpoint request replace `[API_KEY]` with your account API key provided by StreamGuys and replace `[SCRAPER_UUID]` with the UUID for the metadata scraper you wish to retrieve metadata from.

Endpoint: Current Metadata

Send a GET request to the following endpoint to request a JSON payload containing the single most recent metadata object:

```
https://jeta.streamguys.com:8444/[API_KEY]/scraper/[SCRAPER_UUID]/metadata
```

EXAMPLE RESPONSE:

```
{"StreamTitle":"Vicky Emerson - Flight","date":"Thu Dec 08 2016 13:42:21 GMT-0600 (CST)","timestamp":1481226141687}
```

The response JSON object will have the following properties:

Property Name: StreamTitle

Type: String

Description: The most recent in-stream metadata string recorded by the metadata scraper. This string will contain all metadata from the streaming server's StreamTitle field.

Example: Vicky Emerson - Flight

Property Name: `StreamUrl`

Type: `String` or `Undefined`

Description: The most recent `StreamUrl` metadata string recorded by the metadata scraper. This string will contain all metadata from the streaming server's `StreamUrl` field. If the stream does not receive `StreamUrl` metadata from the encoder, then this field will not be present in the JSON response to the API request.

Example:

```
?autoID=OM39&autoCat=OM1&sec=130&dur=125&cat=music&album=&Label=
&YearSG=&Field1=&Field2=
```

Property Name: `date`

Type: `String`

Description: The date of when the metadata changed. This property is localized to SGmetadata server time (Central Standard Time).

Example: `Thu Dec 08 2016 13:42:21 GMT-0600 (CST)`

Property Name: `timestamp`

Type: `Integer`

Description: The epoch timestamp (in milliseconds) of when the metadata changed. This property can be used to build a date string localized to the client's time zone.

Example: `1481226141687`

Endpoint: All Metadata

Send a GET request to the following endpoint to request a JSON payload containing an array of all metadata objects for the scraper:

```
https://jeta.streamguys.com:8444/[API_KEY]/scraper/[SCRAPER_UUID]/metadata/all
```

EXAMPLE RESPONSE:

```
[{"StreamTitle":"Vicky Emerson - Flight","date":"Thu Dec 08 2016 13:42:21
GMT-0600 (CST)","timestamp":1481226141687},{ "StreamTitle":"Beatrix Becker -
```

```
La Gatita Loca","date":"Thu Dec 08 2016 13:39:39 GMT-0600
(CST)","timestamp":1481225979059},{ "StreamTitle":"Whisperings - Customize
Your Playlist with the Whisperings Player...","date":"Thu Dec 08 2016
13:38:16 GMT-0600
(CST)","timestamp":1481225896438},{ "StreamTitle":"Whisperings - Get Your
Copy! Whisperings Solo Piano Vol. 1 and 2","date":"Thu Dec 08 2016 13:37:06
GMT-0600 (CST)","timestamp":1481225826913},{ "StreamTitle":"Catherine Marie
Charlton - Shenandoah (Folk Song)","date":"Thu Dec 08 2016 13:29:57 GMT-0600
(CST)","timestamp":1481225397561},{ "StreamTitle":"Christopher Boscole -
Takako","date":"Thu Dec 08 2016 13:23:39 GMT-0600
(CST)","timestamp":1481225019114}]
```

Each object in the response JSON array will have the same properties as the single response from the [Current Metadata](#) endpoint.

Endpoint: Current Raw Metadata

Send a GET request to the following endpoint to request a single string response containing the most recent unprocessed metadata string directly from the stream. If the encoder is receiving StreamUrl metadata, this endpoint will return a string response containing both the StreamTitle field and the StreamUrl field:

```
https://jeta.streamguys.com:8444/[API_KEY]/scraper/[SCRAPER_UUID]/metadata/raw
```

EXAMPLE RESPONSE:

```
"StreamTitle='Vicky Emerson - Flight';"
```

EXAMPLE RESPONSE WITH STREAMURL:

```
"StreamTitle='JOHN MELLENCAMP - Paper In
Fire';StreamUrl='?autoID=OY49&autoCat=OM1&sec=227&dur=205&cat=mu
sic&album=&Label=&YearSG=&Field1=&Field2=';"
```

Endpoint: All Raw Metadata

Send a GET request to the below endpoint to request a JSON payload containing an array of all metadata strings for the scraper.

```
https://jeta.streamguys.com:8444/[API_KEY]/scraper/[SCRAPER_UUID]/metadata/raw/all
```

EXAMPLE RESPONSE:

```
[{"StreamTitle":"'Vicky Emerson - Flight';", "StreamTitle":"'Beatrix  
Becker - La Gatita Loca';", "StreamTitle":"'Whisperings - Customize  
Your Playlist with the Whisperings  
Player...';", "StreamTitle":"'Whisperings - Get Your Copy!  
Whisperings Solo Piano Vol. 1 and 2';", "StreamTitle":"'Catherine  
Marie Charlton - Shenandoah (Folk  
Song)';", "StreamTitle":"'Christopher Boscole - Takako';"]
```

Each entry in the JSON response array is a raw metadata string with the same formatting as metadata strings returned from the [Current Raw Metadata](#) endpoint.

SOCKET INTEGRATION PROCEDURE

Perform the following steps to add SGmetadata push support to your page. For more detailed integration directions, see the SGmetadata Socket API entries for each function call and event registration listed in the steps below as well as the SGmetadata Socket Integration example section.

- 1) Include socket.io client JavaScript on your page. See the [Socket.io Client Integration](#) section.
- 2) Create a socket by calling `io.connect()`
- 3) Register a `socket.on('connect')` event handler function that contains a call to `socket.emit('listen', scraperUuid)`. This will set your socket to listen for metadata events from your scraper once it is connected.
- 4) Register a `socket.on('metadata')` event handler function. This function will be called every time new metadata for your scraper is pushed through the socket connection. This handler function is where you will place your application logic that deals with metadata sent from the server.
- 5) (Optionally) Register a `socket.on('disconnect')` event handler function. This function will be called every time the socket is disconnected from the server.

SOCKET.IO CLIENT INTEGRATION

Before your page can connect to the SGmetadata server, the socket.io client JavaScript file must be included in your page's head tag. Add the following script tag to your page's head tag:

```
<script type="text/javascript"
src="http://jetm.streamguys.com:8000/socket.io/socket.io.js"></script>
```

This is the Socket.io client JavaScript served by the SGmetadata system's servers.

If you would rather host the Socket.io client JavaScript yourself, please perform the integration directions on the *"Getting the Socket.io Client"* section of Socket.io's download page: <http://socket.io/download/>

SGMETADATA SOCKET API

The SGmetadata socket api consists of functions, socket emit channels, and events. For a complete list of socket.io client functions see: <http://socket.io/docs/client-api/>
The socket.io calls used by SGmetadata are listed below:

SOCKET.IO FUNCTIONS

For a complete list of socket.io client functions see: <http://socket.io/docs/client-api/>
The socket.io calls used by SGmetadata are listed below:

Function: io.connect()

This function sets up the socket.io connection to the SGmetadata server. All subsequent SGmetadata event bindings will be attached to the socket object returned by calling `io.connect()`.

PARAMETERS:

1) **(String) url** – The socket server url string to connect to. For HTTP pages, use:

<http://jetm.streamguys.com:8000>

For HTTPS pages use: <https://jetm.streamguys.com:8443>

2) **(Object) options** – socket.io configuration options object. See socket.io's client-api page for complete documentation of available options. SGmetadata requires at least the 2 following options:

```
{ 'forceNew': true, 'query': 'key='+YOUR_API_KEY }
```

RETURNS:

(Object) socket – A socket.io socket instance. SGmetadata event listeners will be registered to this.

Function: `socket.disconnect()`

This function terminates the socket connection to the SGmetadata server. It should be called on the `socket` object returned by the initial call to `io.connect()`.

This function has no parameters and returns nothing.

SOCKET EMIT CHANNELS

Use these channels with `socket.emit()` calls to send messages to the SGmetadata server over a connected socket.

Channel: `listen`

Call `socket.emit('listen', uuid)` after the `socket.on('connect')` listener has called back to begin listening for metadata events from a scraper matching the given UUID. The first emit of the `listen` channel on a socket will trigger the server to call the `socket.metadata` event with the scraper's current metadata object. Every time

scraper metadata changes after `listen` has been emitted, the `socket metadata` event will be triggered with new scraper metadata from the server.

PARAMETERS:

- 1) **(String) uuid** – The UUID string of the metadata scraper you wish to listen to metadata from.
- 2) **(optional)(Boolean) raw** – Set this to true if you want 'raw' metadata from the server. Raw metadata is not run through server-side special character decoding. In most cases this should be omitted or set to false.

Channel: stop

Call `socket.emit('stop', uuid)` on a connected socket to stop listening to metadata events from the scraper matching the given UUID. This call may be followed up with a new `socket.emit('listen', newUuid)` call if you wish to then listen to metadata from a different scraper.

PARAMETERS:

(String) uuid – The UUID string to stop listening to.

Channel: getMetadata

Call `socket.emit('getMetadata', uuid)` to manually trigger a `metadata` event from the server. Use this when you wish to fetch metadata before it has been pushed from the server. This function should not be used in the most common use cases.

PARAMETERS:

- 1) **(String) uuid** – The UUID string of the metadata scraper you wish to get metadata from.

- 2) **(optional)(Boolean) raw** – Set this to true if you want 'raw' metadata from the server. Raw metadata is not run through server-side special character decoding. In most cases this should be omitted or set to false.

Channel: getScraperInfo

Call `socket.emit('getScraperInfo', uuid)` to trigger a `scraperInfo` event from the server. After calling this emit, a `scraperInfo` object will be sent to your `socket.on('scraperInfo')` event handler function. `scraperInfo` objects contain non-metadata related information about the scraper process.

PARAMETERS:

(String) uuid – The UUID string of a specific scraper.

SOCKET EVENTS

Use these calls with `socket.on('EVENT_NAME', function(params) {
//Handler code here });` to listen for SGmetadata response events.

Event: connect

This event is fired when the socket connects to the SGmetadata server. Place any calls that require an open socket connection (such as `listen` emits) inside the `socket.on('connect')` event handler function.

PARAMETERS:

function() – The function to execute when the connect event is fired. You can emit signals to the socket server inside this connect handler function.

EXAMPLE:

```
//When the socket connects, emit the listen event
//to receive metadata for our specific scraper.
socket.on('connect', function() {
    //Listen for on metadata events
    //for our specific scraper.
    socket.emit('listen', scraperUUID);
});
```

Event: metadata

This event is fired when new scraper metadata is pushed from the server. A `listen` emit must be sent to the server before the `socket.on('metadata')` event will be fired.

PARAMETERS:

function(metadata) – The function to execute when the socket metadata event is fired. The metadata array passed to this function is an array of metadata objects with the following parameters: StreamTitle, StreamUrl, date, timestamp.

EXAMPLE:

```
//Register our main socket metadata event handler.
//When our scraper's metadata changes, it will be
//pushed to this client through this event handler.
socket.on('metadata', function(metadata) {
    //Do something with the metadata object from the server.
    console.log("New metadata received from the server: ", metadata);
});
```

Event: rawMetadata

This event is fired when new scraper raw metadata is pushed from the server. A `listen` emit with the raw Boolean set to true, must be sent to the server before the `socket.on('rawMetadata')` event will be fired.

PARAMETERS:

function(rawMetadata) – The function to execute when the socket rawMetadata event is fired. The rawMetadata array passed to this function is an array of unprocessed metadata strings.

Event: scraperInfo

This event is called when a new `scraperInfo` object is pushed from the server in response to a `scraperInfo` emit.

PARAMETERS:

function(scraperInfo) – The function to execute when the socket scraperInfo event is fired. The scraperInfo object passed to this function contains various fields about the state of the scraper process as well as the url of the stream being scraped for metadata.

Event: disconnect

This event is called when a scraper connection is disconnected from the server.

PARAMETERS:

function(error) – The function to execute when the socket disconnect event is fired. If there was an error that caused the disconnect, it will be passed as an error object to this function. Otherwise, error will be undefined or null.

SGMETADATA SOCKET INTEGRATION EXAMPLE

Place the following JavaScript tag on your page just before its closing body tag (`</body>`). Make sure to set the `apiKey` and `scraperUUID` variables in the example below to the values you received from StreamGuys technical support.

```
<script type="text/javascript">
  //Paste your unique SGmetadata API key here.
  var apiKey = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';

  //Paste your scraper's UUID here.
  var scraperUUID = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx';

  //This is the SGmetadata socket connection url.
  var sgMetadataServer = 'http://jetm.streamguys.com:8000';

  //Our Socket.io connection options.
  //For more connection options see: http://socket.io/docs/client-api/
  var socketOptions = {
    'forceNew':true,
    'query':'key='+apiKey
  };

  //Connect to the SGmetadata socket.
  var socket = io.connect(sgMetadataServer, socketOptions);

  //When the socket connects, emit the listen event
  //to receive metadata for our specific scraper.
  socket.on('connect', function() {
    //Listen for on metadata events
    //for our specific scraper.
    socket.emit('listen', scraperUUID);
  });

  //Register our main socket metadata event handler.
  //When our scraper's metadata changes, it will be
  //pushed to this client through this event handler.
  socket.on('metadata', function(metadata) {
    //Do something with the metadata object from the server.
    console.log("New metadata received from the server: ", metadata);
  });

  //If the socket's connection is interrupted
  //or terminated, this event will be thrown.
  socket.on('disconnect', function(error) {
    console.error('Disconnected from socket server.', error);
  });
</script>
```