

Brief for HTTP Commands for KOD3000

*It is an implementation of a **RESTful service** on our karaoke player, using JSON object as data exchange format.*

| KOD-3000 Host | <-----> APP (i.e. Mobile devices/PC/any network connected client device)

IP:192.168.1.235

If the IP of KOD-3000 player is 192.168.1.235, the HTTP command description as below:

Get the model name command

<http://192.168.1.235/machine.cgi>

Return value:

```
{ "OK": { "type": "KOD3000" } }
```

Get the current playlist command

http://192.168.1.235/songq_list.cgi

Return value sample:

```
{ "OK": { "snos": [ 16858, 83339, 325294 ], "nsongs": 3 } }
```

It means that there are 3 songs (song number:16858,83339, and 325294) in the playlist

Insert the song number 16590 to the top of the playlist

http://192.168.1.235/songq_ins.cgi?sno=16590&pos=0

Return value:

```
{ "OK": { "pos": 0 } }
```

Pause Command

<http://192.168.1.235/pause.cgi>

Return value:

```
{ "OK": {} }
```

Next Command

<http://192.168.1.235/next.cgi>

Return Value:

if exists more songs,

```
{ "OK": {} }
```

if there is no any song in the playlist

```
{ "ERROR": "song queue is empty" }
```

Repeat Command

<http://192.168.1.235/repeat.cgi>

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```

Turn up/down the volume command

the APIs is a relative index adjusting,we just provide a step (+1,-1) adjusting method, for the sake of the same as remote control.

Turn up the volume command volume+1

<http://192.168.1.235/adjustd.cgi?cate=volume&val=1>

(if the current volume is 90 in the player, and you send this sample command, the volume will become 95)

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```

Turn down the volume command Volume-1

<http://192.168.1.235/adjustd.cgi?cate=volume&val=-1>

(if the current volume is 90 in the player, and you send this sample command, the volume will become 85)

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```

Turn up/down the volume of MIC

the APIs is a relative index adjusting, we just provide a step (+1,-1) adjusting method, for the sake of the same as remote control.

Turn up the volume of MIC mic+1

<http://192.168.1.235/adjustd.cgi?cate=mic&val=1&id=0>

Set the mic id,(id 0 means MIC1, id 1 means MIC2)

for example, if you wanna adjust MIC2, the command as below

<http://192.168.1.235/adjustd.cgi?cate=mic&val=1&id=1>

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```

Turn down the volume of MIC mic-1

<http://192.168.1.235/adjustd.cgi?cate=mic&val=-1&id=0>

turn down the volume of MIC1

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```

Adjusting Audio MPX command

Audiompx+1

<http://192.168.1.235/adjustd.cgi?cate=audiompx&val=1>

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```

* audiompx-1

<http://192.168.1.235/adjustd.cgi?cate=audiompx&val=-1>

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```

Toggle to mute/umute command

<http://192.168.1.235/adjustd.cgi?cate=mute&val=1>

Return value:

If it is playing a song

```
{ "OK": {} }
```

If it isn't playing a song

```
{ "ERROR": "no playing song" }
```
