---

title: Using a device controller weight: 45

Typically, you use the local speaker and microphone of the audio device to send and receive audio. However, you might have a smart speaker in your environment that you would like to use to perform external processing or you might have additional device controls, for example, for volume or display. Watson Assistant Solutions provides you with the option to use your own smart speaker and microphone with the audio client. The smart speaker acts as a device controller for the audio client.

Socket interfaces

The audio client provides socket interfaces to allow control of the audio client from a device controller and sending audio to and from the controller. There are two socket interfaces:

- **Command socket interface**: The device controller sends text-based commands on the command socket interface to the audio client. The audio client sends status messages and commands to the device controller. The port is set using the `cmdSocketPort` property in the `configure.properties` file.
- **Audio socket interface**: Audio is sent from the device controller to the client and from the client to the controller on an audio socket interface. The port is set using the `audioSocketPort` property in the `configure.properties` file.

For example, the device controller wants to send audio to the audio client on the audio socket interface, but the client is not ready to process the audio. The device controller sends a RAS (a trigger to the client to read from the audio socket) command on the command socket interface. The audio client responds with a `micWakeUpNotAllowed` status message. The device controller waits for a `micWakeUpAllowed` status message and sends data to the audio client on the audio socket interface.

Command socket interface

You can test the command socket interface using telnet. Complete these steps:

1. Establish a telnet connection to the audio client on the command port.
2. Wait for the client to respond with OK.
3. Send a command and terminate the command with a carriage return.

Table 1 displays the commands from the device controller to the client.

Command	Description
`OS`	Send output to the speaker.
`OAS`	Send output to the audio socket.
`RM`	Read the microphone (trigger).
`RAS`	Read the audio socket (trigger).
`EXIT`	Disconnect.

Table 2 displays the responses from the audio client to the device controller.

Command	Description
`OK`	Command was received and is acknowledged.
`?`	An unknown command was received.
`DONE`	The client was told to disconnect.
`micWakeUpNotAllowed`	The client will not respond to the wake up command trigger.
`micWakeUpAllowed`	The client will respond to the wake up command trigger.
`micOn`	The client is expecting audio.
`micOff`	The client is not expecting audio.

Table 3 displays the status messages from the audio client to the device controller to show the status of the connection to the audio gateway.

Command	Description
`serverConnected`	The client is connected to the server but is not yet ready to start.
`serverConnecting`	The client is attempting to connect to the server.
`serverConnectionReady`	The client is connected and is ready to start.
`serverNotConnected`	The client is not connected to the server.

Audio socket interface

The audio socket interface sends and receives audio data streams in binary format. Currently, the format is fixed as follows:

- Input: audio/l16 (PCM, SampleRate=16,000, Channels=1, Bits=16)
- Output: audio/l16 (PCM, SampleRate=16,000, Channels=1, Bits=16)

If the controller sends an OAS command for diverting audio output to the audio socket, the controller must be ready to receive audio data and process it (that is, play it). No command is sent from the client to indicate that audio data will be sent.

When the controller sends a RAS command to trigger the reading of audio data from the audio socket, the client responds with a `micOn` response and starts to read data from the audio socket. The audio data is sent to the Watson server for transcription. Once the transcription has responded with an acceptable confidence level, the client sends a `micClose` response. Any further data that is received on the audio socket is discarded.

For information about the flow of audio from the audio client to the audio gateway component, see the *How audio input is processed with a controller* topic in the product documentation.

> **What to do next?**
> Learn how audio is processed with a controller.