Creating a Custom Control Module

A CasaTunes Control Module lets CasaTunes automatically send commands to a third-party device — an AV receiver, projector, lighting controller, or any other hardware — whenever a zone changes state (power on/off, volume, mute, source). You define the device's commands in an XML file. No programming is required.

How It Works

When you assign a Control Module to a zone, CasaTunes watches for changes in that zone and sends the appropriate commands to your device over the connection you configure (serial, network, UDP, or HTTP). For example:

- Zone powers on → send `PWR ON` to the AV receiver
- Volume changes → send the new volume level to the amplifier
- Source changes → switch the projector to the correct input

XML File Structure

A control module file has three main sections:

```
<Module Name="Denon1 (Main Room) - Generic Denon Receiver using IP Control"
        Type="AVR" Make="Denon" Unit="1"
        ID="91D9CA35-3A9D-4B85-987C-C0D0FC709F38"
        Comment="Supports Denon AVR-X series">

  <Variables>
    <!-- Optional user-configurable settings -->
  </Variables>

  <Protocols>
    <!-- How to connect to the device -->
  </Protocols>

  <Events>
    <!-- What commands to send and when -->
  </Events>

</Module>
```

Module Attributes

Attribute	Required	Description
`Name`	Yes	Display name shown in the CasaTunes admin UI (see naming convention below)
`ID`	Yes	Unique identifier for this module — must be unique across all modules. Use a GUID. This is the value used to assign the module to a zone.
`Type`	No	Device category, e.g. `AVR` for AV receivers. Used to group modules by type in the UI.
`Make`	No	Manufacturer and model family, e.g. `Cambridge 651/751`. Displayed in the UI.
`Unit`	No	Module number when you have more than one unit of the same device, e.g. `1`, `2`.
`Zone`	No	Zone number for AV receivers that support multiple independent zones, e.g. `1` for the main zone, `2` for Zone 2.
`Comment`	No	Free-text description of supported device models.
`FirstRoomSourceID`	No	Offset added to source IDs before sending (use if device sources are 1-indexed).

Name Convention

Follow this pattern for the `Name` attribute so that modules are easy to identify in the UI:

```
<Make><Unit> (<Main Room | Zone N>) – <Description> using <IP | Serial/RS-232 | UDP> Control
```

Examples:

Scenario	Name
Single AVR, main zone, IP control	`Denon1 (Main Room) – Generic Denon Receiver using IP Control`
Single AVR, second zone, IP control	`Denon1 (Zone 2) – Generic Denon Receiver using IP Control`
Two units of the same model	`Cambridge1 (Main Room) – Cambridge AZUR 651/751 AV Receiver using Serial/RS-232 Control`
Projector, serial control	`Sony1 (Main Room) – Sony VPL Projector using Serial/RS-232 Control`

Generating a unique ID

The `ID` must be a GUID that is unique across all your modules. You can generate one at the command line:

Windows PowerShell:

```
[guid]::NewGuid()
```

Online: search for "GUID generator" and use any reputable tool.

Paste the result (without braces) into the `ID` attribute, for example:

```
ID="91D9CA35-3A9D-4B85-987C-C0D0FC709F38"
```

Protocols

The `<Protocols>` section defines how CasaTunes connects to the device. You can define more than one protocol if a device requires different connection types for different commands.

Serial (RS-232)

```
<Protocol Name="Serial1" Type="Serial"
          COMMPort="3"
          Baud="9600"
          DataBits="8"
          StopBits="One"
          Parity="None"
          Handshake="None"/>
```

Attribute	Description
`COMMPort`	COM port number (e.g. `3` for COM3)
`Baud`	Baud rate (e.g. `9600`, `19200`, `38400`, `115200`)
`DataBits`	Typically `8`
`StopBits`	`One`, `Two`, or `OnePointFive`
`Parity`	`None`, `Odd`, or `Even`
`Handshake`	`None`, `XOnXOff`, `RequestToSend`, or `RequestToSendXOnXOff`

IP / Telnet (TCP/IP)

```
<Protocol Name="IP1" Type="IP"
          IPAddress="192.168.1.100"
```

```
          Port="23"/>
```

CasaTunes opens a TCP connection to the device and keeps it open (with a 1-minute idle timeout).

UDP

```
<Protocol Name="UDP1" Type="UDP"
          IPAddress="192.168.1.100"
          Port="5000"/>
```

UDP is fire-and-forget — there is no acknowledgement. Use it only if the device requires UDP control.

REST (HTTP/HTTPS)

```
<Protocol Name="REST1" Type="REST"
          IPAddress="192.168.1.100"
          Port="80"
          BaseURL="/api"
          UserName="admin"
          Password="secret"/>
```

Attribute	Description
`IPAddress`	Hostname or IP address of the device
`Port`	HTTP port (typically `80` or `443`)
`BaseURL`	Path prefix prepended to every request URI
`UserName` / `Password`	Optional HTTP Basic Auth credentials

Events

The `<Events>` section maps zone state changes to commands. Each `<Event>` contains one or more `<Command>` elements that are sent in order.

Available Event Names

Event Name	When it fires
`Power On`	Zone is powered on
`Power Off`	Zone is powered off
`Mute On`	Zone is muted
`Mute Off`	Zone is unmuted

Event Name	When it fires
`Volume Up`	Volume is increased
`Volume Down`	Volume is decreased
`Volume Changed`	Volume changes to a specific value
`Source Changed`	Zone switches to a different source input
`Matrix Zone On`	A zone on a matrix amplifier is turned on
`All Matrix Zones Off`	All zones on a matrix amplifier are off

You do not need to include all events — only add the ones your device needs.

Command Attributes

```
<Command Name="PowerOn"
         ProtocolName="Serial1"
         DataType="ASCII"
         PreCommandDelay="100"
         PostCommandDelay="200"
         TerminateDataWith="CR"
         ClosePort="false">
  <Data>PWR ON</Data>
</Command>
```

Attribute	Description
`Name`	Descriptive label (for your reference only)
`ProtocolName`	Must match a `Name` in the `<Protocols>` section
`DataType`	How the command data is encoded (see below)
`PreCommandDelay`	Milliseconds to wait **before** sending this command
`PostCommandDelay`	Milliseconds to wait **after** sending this command
`TerminateDataWith`	Terminator appended to the data: `None`, `CR`, `LF`, or `CRLF`
`ClosePort`	`true` to disconnect after this command; `false` (default) to keep the connection open
`Min`, `Max`	For volume commands: scale zone volume (0–100) to the device's range (see Volume below)

DataType Options

`DataType`	Use when
`ASCII`	Device expects plain text commands

DataType	Use when
Hex	Device expects binary data — enter as comma-separated hex bytes: `0x01,0x02,0xAB`
XML	Device expects an XML payload
UrlEncoded	Device expects a URL-encoded form body (REST only)

REST-Specific Command Attributes

```
<Command Name="SetSource"
         ProtocolName="REST1"
         DataType="UrlEncoded"
         RestVerb="POST"
         RestURI="/input/select"
         RestContentType="application/x-www-form-urlencoded">
  <Data>input=%Source:d%</Data>
</Command>
```

Attribute	Description
RestVerb	HTTP method: GET, POST, PUT, or DELETE
RestURI	Path appended to the protocol's BaseURL to form the full request path
RestContentType	Content-Type header for POST/PUT requests

> **BaseURL + RestURI**: The full URL for each request is built from two parts — BaseURL is set once on the <Protocol> element and acts as a common prefix, while RestURI is set per command and provides the specific path. For example, with BaseURL="/api" and RestURI="/input/select", CasaTunes sends the request to http://192.168.1.100:80/api/input/select. If all your commands share the same base path, put it in BaseURL to avoid repeating it on every command.

Substitution Tokens

Use these tokens inside <Data> to insert dynamic values at the time the command is sent:

Token	Replaced with
%Zone:d%	Zone number (decimal)
%Zone:D2%	Zone number (decimal, zero-padded to 2 digits, e.g. 03)
%Zone:x%	Zone number (lowercase hex)
%Zone:X2%	Zone number (uppercase hex, 2 digits)
%Source:d%	Source number (decimal)

Token	Replaced with
`%Source:X2%`	Source number (uppercase hex, 2 digits)
`%Volume:d%`	Volume value (decimal)
`%Volume:x%`	Volume value (hex)
`%VariableName%`	Current value of a `<Variable>` you defined

Examples:

```
<Data>ZONE%Zone:D2% ON</Data>        <!-- Sends e.g. "ZONE02 ON" -->
<Data>VOL %Volume:d%</Data>          <!-- Sends e.g. "VOL 42" -->
<Data>INPUT%Source:d%</Data>         <!-- Sends e.g. "INPUT3" -->
```

Volume Control

Relative volume (Volume Up / Volume Down)

Use `Volume Up` and `Volume Down` events. The device receives a fixed increment command each time the user changes the volume.

```
<Event Name="Volume Up">
  <Command Name="VolUp" ProtocolName="Serial1" DataType="ASCII"
TerminateDataWith="CR">
    <Data>VOL+</Data>
  </Command>
</Event>

<Event Name="Volume Down">
  <Command Name="VolDown" ProtocolName="Serial1" DataType="ASCII"
TerminateDataWith="CR">
    <Data>VOL-</Data>
  </Command>
</Event>
```

Absolute volume (Volume Changed)

Use a `Volume Changed` event and set `Min` and `Max` to match your device's volume range. CasaTunes scales the zone volume (0–100) to that range automatically.

```
<Event Name="Volume Changed">
  <Command Name="SetVol" ProtocolName="IP1" DataType="ASCII"
           Min="0" Max="100" TerminateDataWith="CR">
    <Data>VOL %Volume:d%</Data>
```

```
  </Command>
</Event>
```

If your device uses a 0–50 range, set `Min="0" Max="50"` and CasaTunes will map the zone volume accordingly.

Variables

Variables let you expose a drop-down option to the user in the CasaTunes admin UI — for example, to select which HDMI input a projector should switch to when the zone turns on.

```
<Variables>
  <Variable Name="VideoInput" Title="Video Input" SelectedValue="HDMI1">
    <Option Title="HDMI 1">HDMI1</Option>
    <Option Title="HDMI 2">HDMI2</Option>
    <Option Title="Component">COMP</Option>
  </Variable>
</Variables>
```

Reference the variable in a command using `%VariableName%`:

```
<Event Name="Power On">
  <Command Name="SelectInput" ProtocolName="IP1" DataType="ASCII"
TerminateDataWith="CR">
    <Data>INPUT %VideoInput%</Data>
  </Command>
</Event>
```

The user can change the selected value in the CasaTunes admin UI without editing the XML file.

You can declare multiple `<Variable>` elements — each one appears as a separate drop-down in the UI and can be referenced independently in any command using `%VariableName%`. The CasaTunes UI currently supports a maximum of 2 variables per module.

```
<Variables>
  <Variable Name="VideoInput" Title="Video Input" SelectedValue="HDMI1">
    <Option Title="Use Current Video Input" />
    <Option Title="HDMI 1">HDMI1</Option>
    <Option Title="HDMI 2">HDMI2</Option>
  </Variable>
  <Variable Name="SurroundMode" Title="Surround Mode"
SelectedValue="STEREO">
    <Option Title="Stereo">STEREO</Option>
    <Option Title="Dolby">DOLBY</Option>
    <Option Title="DTS">DTS</Option>
```

```
  </Variable>
</Variables>
```

Suppressing a command with an empty option

If an `<Option>` has no value (self-closing tag with no text content), selecting it tells CasaTunes to **skip the command entirely** — nothing is sent to the device. This is useful when the action is optional and the user may not always want it to fire.

For example, adding a "Use Current Video Input" option lets the user choose whether CasaTunes should switch the projector's input on power-on, or leave it as-is:

```
<Variables>
  <Variable Name="VideoInput" Title="Video Input" SelectedValue="HDMI1">
    <Option Title="Use Current Video Input" />
    <Option Title="HDMI 1">HDMI1</Option>
    <Option Title="HDMI 2">HDMI2</Option>
    <Option Title="Component">COMP</Option>
  </Variable>
</Variables>
```

When "Use Current Video Input" is selected, the `SelectInput` command is suppressed and the projector's current input is left unchanged. When any other option is selected, the command is sent with the chosen value substituted for `%VideoInput%`.

Complete Examples

Example 1 — Serial AV Receiver

```
<Module Name="Yamaha1 (Main Room) - Generic Yamaha AV Receiver using
Serial/RS-232 Control"
        Type="AVR" Make="Yamaha" Unit="1"
        ID="A1B2C3D4-E5F6-7890-ABCD-EF1234567890"
        Comment="RS-232 control via COM3">

  <Protocols>
    <Protocol Name="Serial1" Type="Serial"
              COMMPort="3" Baud="9600" DataBits="8"
              StopBits="One" Parity="None" Handshake="None"/>
  </Protocols>

  <Events>
    <Event Name="Power On">
      <Command Name="PwrOn" ProtocolName="Serial1" DataType="ASCII"
TerminateDataWith="CR">
        <Data>PWR ON</Data>
      </Command>
```

```
    </Event>

    <Event Name="Power Off">
      <Command Name="PwrOff" ProtocolName="Serial1" DataType="ASCII"
TerminateDataWith="CR">
        <Data>PWR OFF</Data>
      </Command>
    </Event>

    <Event Name="Volume Changed">
      <Command Name="SetVol" ProtocolName="Serial1" DataType="ASCII"
               Min="0" Max="80" TerminateDataWith="CR">
        <Data>VOL %Volume:d%</Data>
      </Command>
    </Event>

    <Event Name="Mute On">
      <Command Name="MuteOn" ProtocolName="Serial1" DataType="ASCII"
TerminateDataWith="CR">
        <Data>MUTE ON</Data>
      </Command>
    </Event>

    <Event Name="Mute Off">
      <Command Name="MuteOff" ProtocolName="Serial1" DataType="ASCII"
TerminateDataWith="CR">
        <Data>MUTE OFF</Data>
      </Command>
    </Event>
  </Events>

</Module>
```

Example 2 — Network-Controlled Projector with Input Selection

```
<Module Name="Sony1 (Main Room) - Sony VPL Projector using IP Control"
        Type="Projector" Make="Sony VPL" Unit="1"
        ID="B2C3D4E5-F6A7-8901-BCDE-F12345678901"
        Comment="IP control with selectable input">

  <Variables>
    <Variable Name="VideoInput" Title="Video Input" SelectedValue="HDMI1">
      <Option Title="HDMI 1">HDMI1</Option>
      <Option Title="HDMI 2">HDMI2</Option>
      <Option Title="VGA">VGA</Option>
    </Variable>
  </Variables>

  <Protocols>
    <Protocol Name="IP1" Type="IP" IPAddress="192.168.1.50" Port="7142"/>
  </Protocols>
```

```
  <Events>
    <Event Name="Power On">
      <Command Name="PwrOn" ProtocolName="IP1" DataType="ASCII"
               PreCommandDelay="500" TerminateDataWith="CR">
        <Data>~0000 1</Data>
      </Command>
      <Command Name="SelectInput" ProtocolName="IP1" DataType="ASCII"
               PreCommandDelay="2000" TerminateDataWith="CR">
        <Data>~0012 %VideoInput%</Data>
      </Command>
    </Event>

    <Event Name="Power Off">
      <Command Name="PwrOff" ProtocolName="IP1" DataType="ASCII"
TerminateDataWith="CR">
        <Data>~0000 0</Data>
      </Command>
    </Event>
  </Events>

</Module>
```

> **Tip:** Use `PreCommandDelay` when a device needs time to be ready before accepting the next command — for example, waiting for a projector to finish powering on before sending an input-select command.

Saving Your Control Module File

Once your XML file is ready, save it to the following folder on the CasaTunes music server:

```
{CasaData}\Modules\Control\
```

Where `{CasaData}` is the CasaTunes data folder, typically:

```
C:\ProgramData\CasaTools\CasaTunes\2.0.0.0\Modules\Control\
```

The file can be accessed from another computer on the network using the shared path:

```
\\<server-name>\casadata\Modules\Control\
```

Name the file using this convention:

```
<manufacturer>-device<#>-<ip|serial>-zone<#>.xml
```

Part	Description
`<manufacturer>`	Manufacturer name in lowercase, e.g. `denon`, `cambridge`, `sony`
`device<#>`	Device number when you have more than one unit of the same model, e.g. `device1`, `device2`
`<ip|serial>`	Connection type: `ip` for network control, `serial` for RS-232
`zone<#>`	Zone number for multi-zone devices, e.g. `zone1`, `zone2`. Omit for single-zone devices.

Examples:

Scenario	Filename
Denon receiver, device 1, serial, zone 2	`denon-device1-serial-zone2.xml`
Cambridge AVR, device 1, IP control, main zone	`cambridge-device1-ip-zone1.xml`
Sony projector, device 1, IP control	`sony-device1-ip.xml`

Loading the Module

Restart CasaTunes from the **CasaTunesX App** to load the new module.