AquaControl Simple Control Integration Guide

Version: 1.0
Last Updated: January 8, 2025

Target Audience

- **Control system programmers** working with Crestron, AMX, Control4, QSC Q-SYS
- **AV system integrators** who need straightforward HTTP control
- **Third-party developers** working with control systems that have limited HTTP client capabilities
- **Professionals** working in closed network environments

Introduction

What is Simple Control?

Simple Control is a special integration mode designed for professional AV control systems that need basic HTTP control without the complexity of cookie-based authentication. It provides direct access to essential audio control functions through simple GET and POST requests.

Why Simple Control Exists

Many professional control systems (Crestron, AMX, Control4, QSC Q-SYS) have limited HTTP client capabilities. Their programming environments may not easily handle cookies, session management, or complex authentication flows. Simple Control eliminates these barriers by providing straightforward HTTP endpoints that work with any system capable of making basic web requests.

Security Model and Requirements

Simple Control is **opt-in** and requires explicit administrator setup:

1. **Administrator Action Required**: An administrator must create a user account named "SimpleControl" in the system
2. **Closed Network Deployment**: Simple Control should only be used on isolated, trusted networks - never on public internet
3. **Conscious Security Trade-off**: Creating the SimpleControl user indicates acceptance of reduced authentication in exchange for integration simplicity

How to Enable Simple Control

1. Access the AquaControl web interface by logging in as an administrator
2. Navigate to **Settings** from the vertical side menu
3. Select the **Security** tab from the horizontal menu
4. Create a new user with the exact username "SimpleControl"
5. Set role to "Admin"
6. Use any password (this account will not be used for login)
7. Once this user exists, `/simplecontrol/*` endpoints will accept requests (they reject requests if this user doesn't exist)

Disabling Simple Control

Simple Control can be instantly disabled by deleting the "SimpleControl" user account. This will cause all `/simplecontrol/*` endpoints to reject requests.

API Reference

Base URL

All Simple Control endpoints use: `http://device-ip:8000/v1.0-beta/simplecontrol/`

GET/SET Endpoint Pairs

Most endpoints come in pairs - GET requests return current values and IDs, SET requests use those IDs to make changes.

Channel Control

Get All Channel Mutes

- **Endpoint**: `GET /v1.0-beta/simplecontrol/workingsettings/dsp/chain/mute/`
- **Purpose**: Retrieve mute status for all audio channels
- **Returns**: Array of channel objects with IDs and mute states
- **Example Response**:

```
{
  "success": true,
  "data": [
    {"id": "InputChannel.1", "muted": false},
    {"id": "OutputChannel.1", "muted": false},
    {"id": "OutputChannel.2", "muted": true}
  ]
}
```

Set Channel Mute

- **Endpoint**: `POST /v1.0-beta/simplecontrol/workingsettings/dsp/chain/mute/{id}`
- **Purpose**: Mute or unmute a specific audio channel
- **Parameters**:
  - `{id}`: Channel ID (get from GET request above)
- **Payload**: `{"muted": boolean}`
- **Example**:
  - URL: `POST /v1.0-beta/simplecontrol/workingsettings/dsp/chain/mute/OutputChannel.1`
  - Payload: `{"muted": true}`

Gain Control

Get All Gain Levels

- **Endpoint**: `GET /v1.0-beta/simplecontrol/workingsettings/dsp/block/gain/level`
- **Purpose**: Retrieve gain levels for all gain blocks
- **Returns**: Array of gain objects with IDs and values

- **Note**: Requires gain blocks to be populated on channels. If no gain blocks are configured, the returned list will be empty.

Set Gain Level

- **Endpoint**: `POST /v1.0-beta/simplecontrol/workingsettings/dsp/block/gain/level{id}`
- **Purpose**: Set gain level for a specific gain block
- **Parameters**:
  - `{id}`: Gain block ID (get from GET request above)
- **Payload**: `{"value": number}`
- **Note**: Only works if gain blocks exist on the channels. Use the GET request above to discover available gain block IDs.

Mixer Control

Get All Mixer Settings

- **Endpoint**: `GET /v1.0-beta/simplecontrol/workingsettings/dsp/mixer/sourceMutesAndLevels`
- **Purpose**: Retrieve all mixer source mutes and levels

Set Mixer Source Mute

- **Endpoint**: `POST /v1.0-beta/simplecontrol/workingsettings/dsp/mixer/sourcemute/{id}`
- **Purpose**: Mute/unmute mixer source
- **Parameters**:
  - `{id}`: Mixer source ID (must end with `.SourceMute`)
- **Payload**: `{"value": "true" | "false"}`

Set Mixer Source Level

- **Endpoint**: `POST /v1.0-beta/simplecontrol/workingsettings/dsp/mixer/sourcelevel/{id}`
- **Purpose**: Set mixer source level
- **Parameters**:
  - `{id}`: Mixer source ID (must end with `.SourceLevel`)
- **Payload**: `{"value": string}`

DCA Control

Get DCA Settings

- **Endpoint**: `GET /v1.0-beta/simplecontrol/workingsettings/virtualDVCA/parameters/mutesAndLeve`
- **Purpose**: Retrieve all DCA mutes and levels
- **Note**: DCA channel count varies by product.

Set DCA Mute

- **Endpoint**: `POST /v1.0-beta/simplecontrol/workingsettings/virtualDVCA/parameters/mute/{id}`
- **Purpose**: Mute/unmute DCA

- **Parameters**:
  - `{id}`: DCA ID (format: `DCAChannel.[0-9]+.Mute`)
- **Payload**: `{"value": boolean | string}`
- **Note**: DCA channel count varies by product.

Set DCA Level

- **Endpoint**: `POST /v1.0-beta/simplecontrol/workingsettings/virtualDVCA/parameters/level/{id}`
- **Purpose**: Set DCA level
- **Parameters**:
  - `{id}`: DCA ID (format: `DCAChannel.[0-9]+.Level`)
- **Payload**: `{"value": number | string}`
- **Note**: DCA channel count varies by product.

System Control

Get Power State

- **Endpoint**: `GET /v1.0-beta/simplecontrol/system/frontPanel/info/power`
- **Purpose**: Get current system power state

Set Power State

- **Endpoint**: `POST /v1.0-beta/simplecontrol/system/frontPanel/info/power`
- **Purpose**: Turn system power on or off
- **Payload**: `{"value": "On" | "Off"}`

Preset Control

Recall Preset

- **Endpoint**: `POST /v1.0-beta/simplecontrol/preset/recall/{id}`
- **Purpose**: Load a preset by ID
- **Parameters**: `{id}`: Preset name/ID (must already exist in the system)
- **Note**: Presets must be created through the main AquaControl interface before they can be recalled via Simple Control

Check Preset Progress

- **Endpoint**: `GET /v1.0-beta/simplecontrol/preset/progress`
- **Purpose**: Check if a preset recall is currently in progress
- **Returns**: Status indicating if preset recall is active

HTTP/Curl Examples

Environment Requirements **These examples are written for: - Git Bash on Windows** (recommended) - **Linux/macOS Terminal - WSL (Windows Subsystem for Linux)**

For Windows Command Prompt users: The multi-line syntax with \ won't work. You'll need to put everything on one line or use PowerShell instead.

Basic Workflow Pattern The typical workflow for Simple Control is:

1. **Discover** available endpoints using GET requests
2. **Extract IDs** from the response
3. **Control** using POST requests with those IDs

Important Note: Simple Control is designed for **control only** - not configuration. All system configuration (creating presets, adding DSP blocks, configuring mixers, etc.) must be done through the main AquaControl web interface first. Simple Control then allows you to recall presets and adjust parameters that already exist.

Example 1: Channel Control **Step 1: Get Available Channels (Git Bash/Linux)**

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/chain/mute/
```

Step 1: Get Available Channels (Windows Command Prompt)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/chain/mute/
```

Response:

```
{
  "success": true,
  "data": [
    {"id": "InputChannel.1", "muted": false},
    {"id": "OutputChannel.1", "muted": false},
    {"id": "OutputChannel.2", "muted": true}
  ]
}
```

Step 2: Mute a Channel (Git Bash/Linux)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/chain/mute/
  -H "Content-Type: application/json" \
  -d '{"muted": true}'
```

Step 2: Mute a Channel (Windows Command Prompt)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/chain/mute/
```

Example 2: DCA Control **Get DCA Settings (Git Bash/Linux)**

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/virtualDVCA/parameters/
```

Get DCA Settings (Windows Command Prompt)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/virtualDVCA/parameters/m
```

Set DCA Mute (Git Bash/Linux)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/virtualDVCA/para
  -H "Content-Type: application/json" \
  -d '{"value": true}'
```

Set DCA Mute (Windows Command Prompt)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/virtualDVCA/para
```

Set DCA Level (Git Bash/Linux)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/virtualDVCA/para
  -H "Content-Type: application/json" \
  -d '{"value": -6.0}'
```

Set DCA Level (Windows Command Prompt)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/virtualDVCA/para
```

Example 3: System Power Control Get Power State (Git Bash/Linux)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/system/frontPanel/info/power
```

Get Power State (Windows Command Prompt)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/system/frontPanel/info/power
```

Turn System On (Git Bash/Linux)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/system/frontPanel/info/power \
  -H "Content-Type: application/json" \
  -d '{"value": "On"}'
```

Turn System On (Windows Command Prompt)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/system/frontPanel/info/power -H
```

Example 4: Preset Control Recall Existing Preset (Git Bash/Linux)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/preset/recall/5
```

Recall Existing Preset (Windows Command Prompt)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/preset/recall/5
```

Response if preset doesn't exist:

```
{
  "statusCode": 422,
  "error": "Unprocessable Entity",
  "message": "Error, a Preset with the name: 5 does not exist"
}
```

Check Preset Progress (Git Bash/Linux)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/preset/progress
```

Check Preset Progress (Windows Command Prompt)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/preset/progress
```

Example 5: Mixer Control Get Mixer Settings (Git Bash/Linux)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/mixer/sourceMutesAnd
```

Get Mixer Settings (Windows Command Prompt)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/mixer/sourceMutesAnd
```

Set Mixer Source Mute (Git Bash/Linux)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/mixer/source
  -H "Content-Type: application/json" \
  -d '{"value": "true"}'
```

Set Mixer Source Mute (Windows Command Prompt)

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/mixer/source
```

Example 6: Gain Control (if gain blocks exist) Get Available Gain Controls (Git Bash/Linux)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/block/gain/level
```

Get Available Gain Controls (Windows Command Prompt)

```
curl http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/block/gain/level
```

Set Gain Level (Git Bash/Linux)

Note: Unlike other endpoints, this one has no slash between "level" and the ID.

Note: The BlockPosition number will vary based on where the gain block is in your DSP chain.

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/block/gain/
  -H "Content-Type: application/json" \
  -d '{"value": -3.0}'
```

Set Gain Level (Windows Command Prompt)

Note: Unlike other endpoints, this one has no slash between "level" and the ID.

Note: The BlockPosition number will vary based on where the gain block is in your DSP chain.

```
curl -X POST http://192.168.1.100:8000/v1.0-beta/simplecontrol/workingsettings/dsp/block/gain/
```

Important Notes for Curl Usage

- **Always include `Content-Type: application/json`** for POST requests
- **Replace `192.168.1.100`** with your actual device IP address
- **Use the exact IDs** returned by GET requests

- **Windows Command Prompt**: Escape quotes with `\"` and put commands on single lines
- **Git Bash**: Use single quotes and backslash for line continuation
- **Check for errors** in the response before proceeding to next operation

---

Python Examples

Environment Requirements **Python Version**: Python 3.6 or higher recommended

Required Library:

```
pip install requests
```

Import Statements: All examples require these imports at the top of your Python script:

```
import requests
import json
```

Basic Workflow Pattern The typical workflow for Simple Control with Python follows the same pattern as curl:

1. **Discover** available endpoints using GET requests
2. **Extract IDs** from the JSON response
3. **Control** using POST requests with those IDs

```
import requests
import json

class AquaControlSimple:
    """Complete Python interface for AquaControl Simple Control API

    NOTE: Simple Control is for control only - not configuration.
    All presets, DSP blocks, mixers, etc. must be created through
    the main AquaControl web interface first.
    """

    def __init__(self, device_ip, port=8000):
        self.base_url = f"http://{device_ip}:{port}/v1.0-beta/simplecontrol"

    def _make_request(self, method, endpoint, payload=None):
        """Internal method for making HTTP requests with error handling"""
        url = f"{self.base_url}{endpoint}"
        try:
            if method.upper() == 'GET':
                response = requests.get(url)
            elif method.upper() == 'POST':
                response = requests.post(url, json=payload)

            response.raise_for_status()
            return response.json() if response.content else True

        except requests.exceptions.ConnectionError:
            print(f"Error: Could not connect to {url}")
```

```
            return None
        except requests.exceptions.HTTPError as e:
            print(f"HTTP Error {response.status_code}: {e}")
            return None
        except requests.exceptions.RequestException as e:
            print(f"Request Error: {e}")
            return None

    # Channel Control Methods
    def get_channels(self):
        """Get all channel mute states"""
        return self._make_request('GET', '/workingsettings/dsp/chain/mute/')

    def set_channel_mute(self, channel_id, muted):
        """Mute or unmute a channel"""
        endpoint = f"/workingsettings/dsp/chain/mute/{channel_id}"
        return self._make_request('POST', endpoint, {"muted": muted})

    # DCA Control Methods
    def get_dca_settings(self):
        """Get all DCA mutes and levels"""
        return self._make_request('GET', '/workingsettings/virtualDVCA/parameters/mutesAndLevel

    def set_dca_mute(self, dca_id, muted):
        """Mute or unmute a DCA channel (e.g., 'DCAChannel.1.Mute')"""
        endpoint = f"/workingsettings/virtualDVCA/parameters/mute/{dca_id}"
        return self._make_request('POST', endpoint, {"value": muted})

    def set_dca_level(self, dca_id, level):
        """Set DCA level (e.g., 'DCAChannel.1.Level', -6.0)"""
        endpoint = f"/workingsettings/virtualDVCA/parameters/level/{dca_id}"
        return self._make_request('POST', endpoint, {"value": level})

    # System Power Control Methods
    def get_power_state(self):
        """Get current system power state"""
        return self._make_request('GET', '/system/frontPanel/info/power')

    def set_power_state(self, power_on=True):
        """Turn system power on or off"""
        power_value = "On" if power_on else "Off"
        return self._make_request('POST', '/system/frontPanel/info/power', {"value": power_valu

    # Preset Control Methods
    def recall_preset(self, preset_id):
        """Recall a preset by ID (preset must already exist in system)"""
        return self._make_request('POST', f'/preset/recall/{preset_id}')
```

```
    def get_preset_progress(self):
        """Check if preset recall is in progress"""
        return self._make_request('GET', '/preset/progress')

    # Mixer Control Methods
    def get_mixer_settings(self):
        """Get all mixer source mutes and levels"""
        return self._make_request('GET', '/workingsettings/dsp/mixer/sourceMutesAndLevels')

    def set_mixer_source_mute(self, source_id, muted):
        """Mute mixer source (e.g., 'Mixer.1.InputChannel.1.SourceMute')"""
        endpoint = f"/workingsettings/dsp/mixer/sourcemute/{source_id}"
        mute_value = "true" if muted else "false"
        return self._make_request('POST', endpoint, {"value": mute_value})

    def set_mixer_source_level(self, source_id, level):
        """Set mixer source level (e.g., 'Mixer.1.InputChannel.1.SourceLevel')"""
        endpoint = f"/workingsettings/dsp/mixer/sourcelevel/{source_id}"
        return self._make_request('POST', endpoint, {"value": str(level)})

    # Gain Control Methods (requires gain blocks to be configured)
    def get_gain_controls(self):
        """Get all available gain level controls"""
        return self._make_request('GET', '/workingsettings/dsp/block/gain/level')

    def set_gain_level(self, gain_id, level):
        """Set gain level (note: no slash before ID in endpoint)"""
        endpoint = f"/workingsettings/dsp/block/gain/level{gain_id}"
        return self._make_request('POST', endpoint, {"value": level})


# Usage Examples
if __name__ == "__main__":
    # Connect to device
    aqua = AquaControlSimple("192.168.1.100")

    print("=== Channel Control ===")
    # Get available channels
    channels = aqua.get_channels()
    if channels:
        print("Available channels:")
        for ch in channels:
            print(f"  {ch['id']}: {'Muted' if ch['mute'] else 'Unmuted'}")

        # Mute first output channel
        if aqua.set_channel_mute("OutputChannel.1", True):
            print("Channel muted successfully")
```

```
print("\n=== DCA Control ===")
# Get DCA settings
dca_settings = aqua.get_dca_settings()
if dca_settings:
    print("DCA Settings:")
    for setting in dca_settings:
        print(f"  {setting['id']}: {setting['value']}")

    # Mute DCA Channel 1
    if aqua.set_dca_mute("DCAChannel.1.Mute", True):
        print("DCA muted successfully")

    # Set DCA level
    if aqua.set_dca_level("DCAChannel.1.Level", -6.0):
        print("DCA level set successfully")

print("\n=== System Power ===")
# Get power state
power_state = aqua.get_power_state()
if power_state:
    print(f"Current power state: {power_state}")

# Turn system on
if aqua.set_power_state(True):
    print("System powered on successfully")

print("\n=== Preset Control ===")
# Recall preset (must exist first - create through main AquaControl interface)
result = aqua.recall_preset("1")
if result:
    print("Preset '1' recalled successfully")
else:
    print("Failed to recall preset '1' (may not exist)")

# Check progress
progress = aqua.get_preset_progress()
if progress is not None:
    print(f"Preset progress: {progress}")

print("\n=== Mixer Control ===")
# Get mixer settings
mixer_settings = aqua.get_mixer_settings()
if mixer_settings:
    print("Mixer settings available")

    # Mute a mixer source
    if aqua.set_mixer_source_mute("Mixer.1.InputChannel.1.SourceMute", True):
```

```
            print("Mixer source muted successfully")

    print("\n=== Gain Control ===")
    # Get gain controls (only if gain blocks exist)
    gain_controls = aqua.get_gain_controls()
    if gain_controls:
        print("Available gain controls:")
        for control in gain_controls:
            print(f"  {control['id']}: {control['value']}")

        # Set gain level (BlockPosition depends on actual configuration)
        if aqua.set_gain_level("InputChannel.1.BlockPosition.2.Level", -3.0):
            print("Gain level set successfully")
    else:
        print("No gain blocks configured")
```

AquaControl Simple Control Python Class

Important Notes for Python Usage

- **Always include proper headers** for POST requests: `{'Content-Type': 'application/json'}`
- **Use `json=payload`** parameter in requests.post() instead of manually encoding JSON
- **Replace `192.168.1.100`** with your actual device IP address
- **Handle exceptions** properly using try/except blocks
- **Use `response.raise_for_status()`** to automatically handle HTTP error codes
- **Check response codes** before processing JSON data
- **Use exact IDs** returned by GET requests for POST operations

---

HTTP Responses and Error Handling

The API returns standard HTTP status codes. Error responses usually include helpful information about what went wrong.

Common HTTP status codes: - **200 OK** - Request successful - **400 Bad Request** - Invalid data or missing required fields - **404 Not Found** - Invalid endpoint or parameter ID - **500 Internal Server Error** - Server error

Check both the HTTP status code and response body for error details when troubleshooting issues.

Preset Recall Blocking Behavior

No Simple Control API calls are allowed while a preset recall is in progress. All requests will be blocked with an error until the preset recall completes.

If you get a blocking error during preset recall: - Poll the `/preset/progress` endpoint every 3 seconds to check status - Wait for preset recall to finish before retrying your request - Optionally

show “preset recall in progress” message or block UI during this time

---

Rate Limiting and Best Practices

Rate Limit: Maximum 5 requests per second to prevent system overload.

Best Practices: - Don’t send rapid-fire requests - space them out appropriately - Poll values once every 3 seconds to keep in sync and provide eventual consistency - Test your control sequences thoroughly before deployment

For high-frequency control applications, consider the rate limit when designing your update intervals.
