PRIME WebSocket Automation Interface Reference Guide

Version 5.3

January 2026

SINCE DAY ONE.

Chyron PRIME WebSocket Automation Interface Reference Guide • 5.3 • January 2026
This document is distributed by Chyron in online (electronic) form only, and is not available for purchase in printed form.

Table of Contents

Interface Specification

API Overview

interface Root

- Runtime Runtime { get; }
- ProjectManager Projects { get; }

interface Runtime

- Switcher Switcher { get; }
- IChannel[] Channels { get; }
- IClipPlayer[] ClipPlayers { get; }

Assets/Projects API

```
enum PathType
        File,
        Directory

interface IName
        string Name { get; }

interface IPath : IName
        PathType Type { get; }
        string Path { get; }

interface IProject : IPath

interface ProjectManager
        /// Returns the current project
        IProject CurrentProject { get; }
        /// Returns a list of all project names (same as available in combobox)
        string[] AllProjects { get; }
        /// Changes the current project for the application
        void SetCurrentProject(string name)
```

Graphics/Clips API

```
enum PlayoutState
        Loading,
        Loaded,
        PlayPending
        Playing,
        Stopping,
        Stopped,
        Closing,
        Closed

interface ISize
        int Width, Height { get; }

interface IScene : IName
        InstanceId { get; }
        ISize Size { get; }
        float FrameRate { get; }
        IBitmap Thumbnail { get; }
        PlayoutState PlayoutState { get; }
        IReplaceable[] Replaceables { get; }
        IControlPanel ControlPanel { get; }
        bool Update(string name, object value);
        event PlayoutStateChanged(PlayoutState playoutState);
        event ReplaceableChanged(string id, string value);

        string[] Actions { get; } // Typically actions are only triggered through button control
                clicks
        void PlayAction(string action)

interface IReplaceable
        string Id { get; }
        string Type { get; } //String, Boolean, Int32, Double
        string Value { get; set; }

interface IPoint
        int X, Y { get; }
```

```
interface IControl : IName
        string Type { get; } // Button, CheckBox, TextBox, ComboBox
        IPoint Location { get; }
        ISize Size { get; }
        object Value  { get; }
        string ForeColor, BackColor { get; } // "#FF7F00" or "#7FFF7F00" with alpha
        string FontName
        float FontSize
        string FontStyle // "Bold", "Italic", or "Bold, Italic"
        IBitmap BackgroundImage // can be null

interface IControlPanel : IControl
        IControl[] Controls { get; }

interface IChannel
        string Name { get; }
        IScene[] OpenScenes { get; }
        IScene GetScene(int instanceId)
        void LoadScene(string path)
        void PlayScene(string path)
        void StopScene(string path)
        void CloseScene(string path)
        void CloseAllScenes()
        event PlayoutStateChanged(int instanceId, PlayoutState playoutState)
        event SceneAdded(IScene scene)
        event SceneRemoved(IScene scene)

enum ClipState
        None
        Cued
        Playing
        Paused

interface IClipScene : IScene
        ClipState ClipState { get; }
        float Rate { get; }
        int TotalFrames { get; }
        int CurrentFrame { get; }
        bool Loop { get; set; }
        event ClipStateChanged(ClipState clipState)
```

```
interface IClipPlayer : IChannel
        IClipScene CuedScene { get; }
        IClipScene PlayingScene { get; }
        void HomeCued()
        void PlayCued()
        void ClearCued()
        void HomePlaying()
        void PlayOrResumePlaying()
        void PausePlaying()
        void StopPlaying()
        void ClearPlaying()
        event ClipStateChanged(int instanceId, ClipState clipState)
```

Switcher API

enum SourceType

- Black
- VideoInput
- ClipPlayer
- Graphic
- Dve
- MixEffect

interface Source

- string Name { get; }
- int Index { get; }
- SourceType Type { get; }

interface Bus

- int Index { get; }
- int Layer { get; } // 0 for Preset, Program, 1-N for Key 1-N
- string Name { get; } // Preset, Program, Key1 - KeyN
- Source ActiveSource { get;}
- bool IsOnProgram { get; }
- event ActiveSourceChanged(Source source);
- event IsOnProgramChanged(bool onProgram);
- void SetActiveSource(string name);

interface AudioBus

- double Volume { get; set; }
- bool IsMuted { get; set; }
- /// Lowers volume by animating towards "scale * Volume"
  ///   scale: applied to current volume to get new volume (0-1)
  ///   frames: animation duration in frames. Optional: defaults to configured audio duck frame count
  void Duck(double scale, int? frames);
- /// Reverts volume to value before Duck
  ///   frames: animation duration in frames. Optional: defaults to configured audio duck frame count
  void Unduck(int? frames);
- /// Fades volume to given level
- ///   targetLevel: fade to this level (0-1)
  ///   frames: duration in frames. Optional: defaults to configured audio duck frame count
  void Fade(double targetLevel, int? frames);

interface KeyBus : Bus

- void Cut()
- void AutoTrans()

interface Transfer

- int Index { get; }
- string Name { get; } // Background, Key1 - KeyN
- bool Enabled { get; set; }
- event EnabledChanged(Transfer transfer)

interface Bank

- string Name { get; }
- Bus[] Buses { get; } // all buses
- Transfer[] Transfers { get; }
- Bus Preset { get; }
- Bus Program { get; }
- Bus[] BackgroundBuses { get; } // Preset and Program
- KeyBus[] KeyBuses { get; }
- void Cut()
- string[] TransitionNames { get; }
- string ActiveTransition { get; }
- event ActiveTransitionChanged(string transitionName)
- void SetActiveTransition(string name)
- void AutoTrans()

- Transfer GetTransfer(string name)
- bool SetTransfer(string name, bool enabled)

interface Switcher

- Source[] Sources { get; }
- Bank[] Banks { get; } // all banks
- Bank Main { get; }
- Bank[] MixEffectBanks { get; }
- AudioBus[] AudioBuses { get; }
- AudioBus GetAudioBus(string name);
- Switcher GetState()

Communication Protocol

Each request is a JSON object containing the following fields:

Id: request identifier (positive integer), used to couple requests and responses within one connection. Always required.
Type: request type, one of following:

- `get`: retrieve value
- `set`: set value
- `call`: call method
- `attach`: add event handler, events are sent containing the id supplied as parameter
- `detach`: remove event handler

Can be omitted, in which case `call` is assumed.
Method: request path relative to connected endpoint. See API for available paths. Each part of the path is separated by a dot. Always required. Arrays can be indexed by placing the index inside parentheses at the end of the path. For function calls, the parentheses at the end must be omitted, they are implied by the `call` type.
Params: parameters for the request. Not allowed for `get` requests.
`set`: exactly one parameter must be supplied. This will be the new value.
`call`: number of parameters must match function parameters.
`attach`: exactly one parameter must be supplied. This can be any id which will be sent in event invocations.
`detach`: any number of parameters can be supplied. Without parameters all handlers will be removed for that path. Any parameter supplied must be an id that was previously attached, which will be detached.

Multiple requests can be sent at once by putting multiple request objects into a JSON array.

Examples

```
//send; connect to switcher: wss://1.2.3.4:5678/Runtime/Switcher
{ id: 1, type: "set", method: "Main.Transfers(0).Enabled", params: [ true ] } //
        Main.Transfers[0].Enabled = true;
{ id: 2, method: "Main.Cut" } // Cut();
{ id: 3, method: "Main.SetActiveTransition", params: [ "Dissolve" ] } //
        SetActiveTransition("Dissolve")
{ id: 9, method: "Main.Preset.SetActiveSource", "params": [ "GFX5" ] } //
        Main.Preset.SetActiveSource("GFX5");
{ id: 4, type: "get", method: "Main.Program.ActiveSource" } // return Program.ActiveSource;
{ id: 5, type: "attach", method: "Main.Preset.ActiveSourceChanged", params: [ 6 ] } //
        Program.ActiveSourceChanged += 6;
{ id: 7, type: "detach", method: "Main.Preset.ActiveSourceChanged", params: [ ] } //
        Program.ActiveSourceChanged -= 6;
{ id: 8, type: "get", method: "Sources(1)" } // return Sources[1];
{ id: 9, type: "call", method: "AudioBuses(0).Duck", params: [0.2] } // AudioBuses[0].Duck(0.2);

//send; connect to bus directly: wss://1.2.3.4:5678/Runtime/Switcher/Main
{ id: 1, type: "set", method: "Transfers(0).Enabled", params: [ true ] } // Transfers[0] = true;
{ id: 2, method: "Cut" } // Cut();
{ id: 3, type: "call", method: "SetActiveTransition", params: [ "Dissolve" ] } //
        SetActiveTransition("Dissolve")
{ id: 9, method: "Preset.SetActiveSource", "params": [ "GFX5" ] } //
        Main.Preset.SetActiveSource("GFX5");
{ id: 4, type: "get", method: "Program.ActiveSource" } // return Program.ActiveSource;
{ id: 5, type: "attach", method: "Preset.ActiveSourceChanged", params: [ 6 ] } //
        Program.ActiveSourceChanged += 6;
{ id: 7, type: "detach", method: "Preset.ActiveSourceChanged", params: [ 6 ] } //
        Program.ActiveSourceChanged -= 6;

// allow array of commands to be sent
[
{ id: 1, type: "get", method: "Preset.Source", params: [ "CAM1" ] },
{ id: 2, method: "Cut" }
]
```

Responses

Each request sent triggers a response. Attached event handlers also send these response objects.

Response Specification

This response is a JSON object containing the following fields:

Id: the request id used to trigger the request. If no id was supplied (invalid request) this will be `-1`. For event handlers this will be the supplied parameter.
Error: an object containing two fields: `Code` (integer) and `Message` (string). Only present if the request fails.
Result: the value returned by a `get` request, omitted for other request types
Params: the event parameters. Only present for event callbacks.

Examples

```
//receive
{ Id: 3 }
{ Id: 4, Result: { Index: 0, Type: "Input", Name: "CAM1" } }
{ Id: 5 }
{ Id: 6, Params: [ { Name: "Program", Source: "CAM2" } ] }
{ Id: 7 }
{ Id: 9, Error: { Code: 123, Message: "Invalid command" } }
```

Tools

The API can easily be tested from the browser using this page:
websocketclient.html

ABOUT US

Chyron is ushering in the next generation of storytelling in the digital age. Founded in 1966, the company pioneered broadcast titling and graphics systems. With a strong foundation built on over 50 years of innovation and efficiency, the name Chyron is synonymous with broadcast graphics. Chyron continues that legacy as a global leader focused on customer-centric broadcast solutions. Today, the company offers production professionals the industry's most comprehensive software portfolio for designing, sharing, and playing live graphics to air with ease. Chyron products are increasingly deployed to empower OTA & OTT workflows and deliver richer, more immersive experiences for audiences and sports fans in the arena, at home, or on the go.

CONTACT SALES

EMEA • North America • Latin America • Asia/Pacific
+1.631.845.2000 • sales@chyron.com

Chyron.