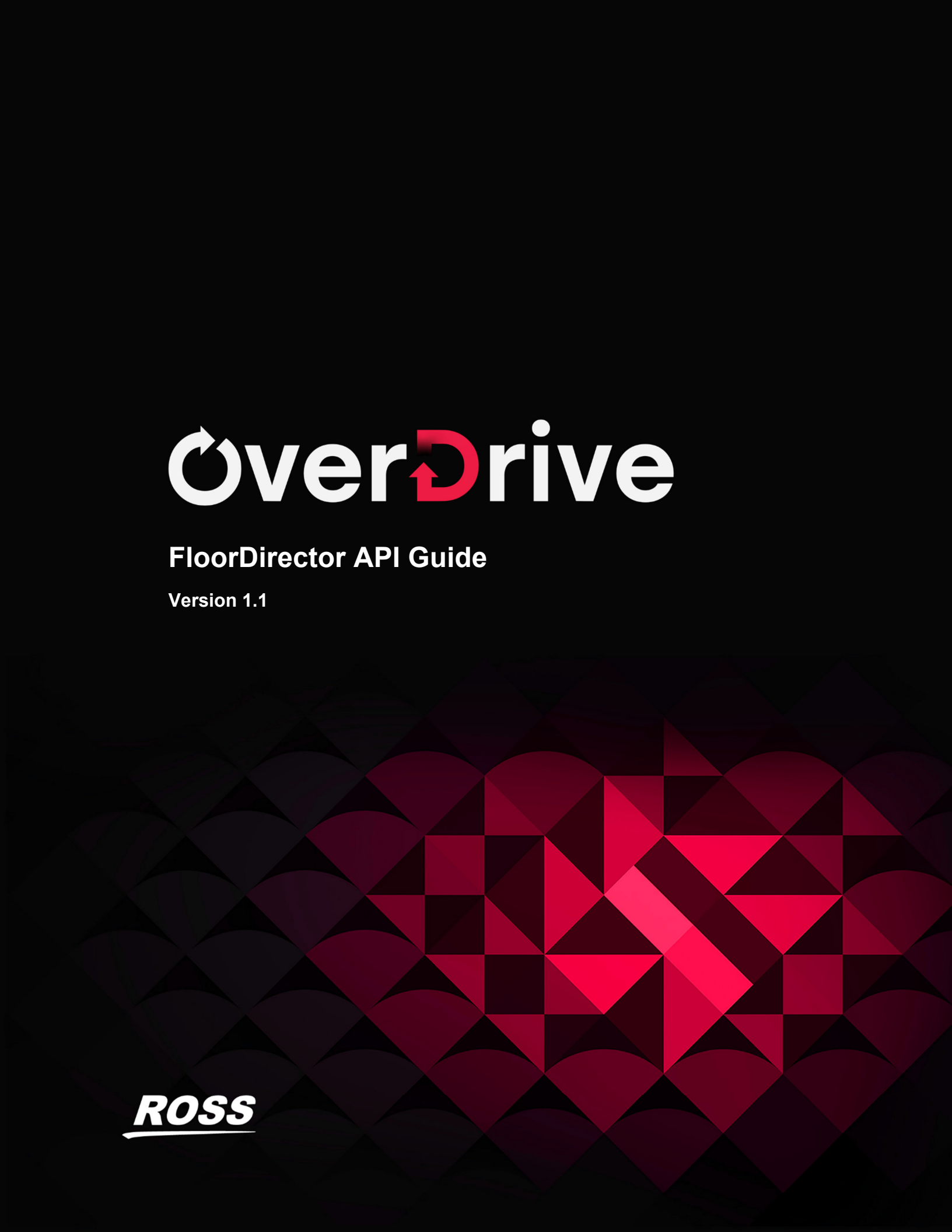
OverDrive

FloorDirector API Guide

Version 1.1



ROSS

Thank You for Choosing Ross

You've made a great choice. We expect you will be very happy with your purchase of Ross Technology.

Our mission is to:

1. Provide a Superior Customer Experience
   - offer the best product quality and support
2. Make Cool Practical Technology
   - develop great products that customers love

Ross has become well known for the Ross Video Code of Ethics. It guides our interactions and empowers our employees. I hope you enjoy reading it below.

If anything at all with your Ross experience does not live up to your expectations be sure to reach out to us at solutions@rossvideo.com.

David Ross

CEO, Ross Video

david.ross@rossvideo.com

Ross Video Code of Ethics

Any company is the sum total of the people that make things happen. At Ross, our employees are a special group. Our employees truly care about doing a great job and delivering a high quality customer experience every day. This code of ethics hangs on the wall of all Ross Video locations to guide our behavior:

1. We will always act in our customers' best interest.
3. We will do our best to understand our customers' requirements.
4. We will not ship crap.
5. We will be great to work with.
6. We will do something extra for our customers, as an apology, when something big goes wrong and it's our fault.
7. We will keep our promises.
8. We will treat the competition with respect.
9. We will cooperate with and help other friendly companies.
10. We will go above and beyond in times of crisis. If there's no one to authorize the required action in times of company or customer crisis - do what you know in your heart is right. (You may rent helicopters if necessary.)

FloorDirector API Guide

- Ross Part Number: 7900DR-010-1.1
- Release Date: January 3, 2025. Printed in Canada.
- Software Issue: 1.1

The information contained in this Guide is subject to change without notice or obligation.

Copyright

Patents

Patent numbers 4,205,346; 5,115,314; 5,280,346; 5,561,404; 7,034,886; 7,508,455; 7,602,446; 7,834,886; 7,914,332; 8307284, 2039277; 1237518; 1127289 and other patents pending.

Notice

The material in this manual is furnished for informational use only. It is subject to change without notice and should not be construed as commitment by Ross Video Limited. Ross Video Limited assumes no responsibility or liability for errors or inaccuracies that may appear in this manual.

Company Address

Ross Video Limited	Ross Video Incorporated
8 John Street	P.O. Box 880
Iroquois, Ontario	Ogdensburg, New York
Canada, K0E 1K0	USA 13669-0880

General Business Office:	(+1) 613 - 652 - 4886
Fax:	(+1) 613 - 652 - 4425
Technical Support:	(+1) 613 - 652 - 4886
After Hours Emergency:	(+1) 613 - 349 - 0006
E-mail (Technical Support):	techsupport@rossvideo.com
E-mail (General Information):	solutions@rossvideo.com
Website:	http://www.rossvideo.com

Contents

Overdrive FloorDirector API Services

Overview

The FloorDirector API will enable users to query timing, shot, cue, and OD system information. JSON responses will be returned for supported queries.

There will be two types of services: RESTful and Socket.

- The RESTful Services will provide the access to the API through http connections.
  - > The simplest one. Easy to test. Can connect even from JavaScript.
  - > No (firewall) security system port rules needed (always port 80)
  - > The request parameters must be provided in a GET Query String.
- The Socket Services will provide the access to the API through a socket connection.
  - > Clients most implement a socket connection.
  - > Clients are responsible for keeping the connection open (OD will send responses and updates through the same socket the client opened). Therefore, clients should monitor their sockets to check the connection health.
  - > The request parameters must be provided in a JSON message.

Although the requests may differ depending on the service type (REST or Socket), the response will be always in the same JSON format described in the API Endpoint Definitions.

API Endpoints (Summary)

The FloorDirector API will expose the following endpoints.

Endpoint	Description
timers	Gets the rundown timing information.
shots	Gets the OnAir/Prepared shot information.
info	Gets the system information (Playing Rundown Name, controlling RC host, active server host, backup server host).
cues	Gets the last ‘cue’ messages.
variable	Gets rundown variable view information (audio variable status).
subscription	**Socket Only** - It subscribes/unsubscribes the current client connection for receiving autonomous timing updates. The default behavior for new connections is being subscribed.

For more endpoint definitions see “API Endpoint Definitions”.

RESTful Services

REST API PATH and Version

The base path will be set to http://<odserver>/server/floordirector/api/<endpoint>

- The version of the API (`v1`) will appear as a prefix to the endpoint.
  - > For example, to get "timers" info, you'd make a GET request to:

    `http://<odserver>/server/floordirector/api/v1/timers?<parameters>`

Parameters

The request parameters must be provided in a GET Query String

For non-ASCII characters:

- Represent each character in UTF-8 (see [RFC2279]) as one or more bytes.
- Escape these bytes with the URI escaping mechanism (i.e., by converting each byte to %HH, where HH is the hexadecimal notation of the byte value).

Also, see:

- https://www.w3.org/TR/html40/appendix/notes.html#non-ascii-chars
- https://en.wikipedia.org/wiki/Query_string

Message Exchange

Because the RESTful API is a server-client service (HTTP), the client always has to pull data from the server.

For timing updates, the client must implement a mechanism to constantly pull data from the server (half a second should be enough).

Socket Services

The interaction can be generated either by the client (**Request-Response**) or by the server (**Broadcast**, IFF the client has already established a socket connection).

Message Transport

Clients must open a socket to the defined Overdrive FloorDirector port. The default TCP/IP port will be set to **8760** (it could be re-configured from the Overdrive WEB Server Configuration page). In order to prevent client applications from exhausting the server resources, there will be a limit of up to 100 open connections to the server. Besides that, to prevent a single client application from exhausting the server connection capacity, there will be a limit of up to 10 open connections per client IP address.

Message Request

All the client requests must be sent in a single message in JSON format.

Request JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "protocolVersion": {
      "type": "string"
    },
    "endpoint": {
      "type": "string"
    },
    "reqData": {
      "type": "string"
    }
    ,
    "correlationId": {
      "type": "string"
    }
  }
}
```

Field Description

Field	Description	Values
protocolVersion	The version of the API.	1
endpoint	API exposed endpoints.	timers, shots, info, cues, variables
reqData	Required Endpoint Parameters, query string format as follows: param1=value1&param2=value2 …	See API Endpoint Definitions
correlationId	Since the server can send multiple asynchronous updates to the client, we recommend the client to include an id in the request, the response to that request will have the same id.	

Example

```
{
  "protocolVersion": "1",
  "endpoint": "timers",
  "reqData": "type=user&timerId=Timer1",
  "correlationId": "1000001"
}
```

Message Exchange

To request updates to the server and receive them:

1. The client application will open a socket on the appropriate port to the OD Server (if a socket has not already been established)
2. The client will then hold the socket open
3. The client application can send the message and wait for any incoming updates on the same socket.

Important Note: Once a client connects and sends the first request message it will automatically receive autonomous updates (see Subscription endpoint). The default timing update frequency will be 500ms but it could be re-configured from the Overdrive WEB Server FloorDirector page.

API Endpoint Definitions

For simplifying the examples, they show only a REST request, see Socket Services for building a socket request.

Timers

Gets rundown timing information (timing sources).

Parameters

URL Parameters	Description	Values	Required
type	The timing source type.	"static" – only the static timers. "user" – only the user timers (per Show). "" – empty for all.	No
timerId	Limits response to only include the named timer id. If omitted, all timers will be included in the response.	Static Timer Names: • "Program Time Elapsed" • "Shot Time Elapsed" • "Story Time Elapsed" • "Clip Time Elapsed (*TYPE-ID*)" • "Clip Time Remaining (*TYPE-ID*)" • "Clock (12 Hour)" • "Clock (24 Hour)" • "NRCS Estimated Duration Remaining" • "NRCS Target Time Remaining" • "NRCS Media Time Remaining" • "NRCS Rundown Start Time Remaining"	No

TYPE-ID: Static clip timers depend on the crosspoints or video servers they are based to. You must specify an extra id to get the corresponding clip timer (as is in RC), add:

- `XPT-<crosspoint>`: When the Clip source is a Crosspoint.
- `VS-<videoserver>`: When the Clip source is a Video Server.

For example:

- To get a static clip timer from (let's say) the crosspoint 24, you would use:
  - > 'timerId=Clip Time Elapsed (XPT-24)'
  - > 'timerId=Clip Time Remaining (XPT-24)'
- To get a static clip timer from a video server named BlackStormVS, you would use:
  - > 'timerId=Clip Time Elapsed (VS- BlackStormVS)'
  - > 'timerId=Clip Time Remaining (VS-BlackStormVS)'

If you request static timers without filters, you will get both all crosspoint-based and all videoserver-based timers.

Response: JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "meta": {
      "type": "object",
      "properties": {
        "serverDate": {
          "type": "string"
        },
        "odHost": {
          "type": "string"
        },
        "odVersion": {
          "type": "string"
        }
        "correlationId": {
          "type": "string"
        }
      }
    },
    "timers": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "timerId": {
            "type": "string"
          },
          " userTimerName ": {
            "type": "string"
          },
          "type": {
            "type": "string"
          },
          "countThrough": {
            "type": "boolean"
          },
          "direction": {
            "type": "string"
          },
          "running": {
            "type": "boolean"
          },
          "duration": {
            "type": "integer"
          },
          "startTime": {
            "type": "string"
          },
          "currentValue": {
            "type": "string"
          },
          "clipName": {
            "type": "string"
          },
          "position": {
            "type": "integer"
          },
          "manualBehaviour": {
            "type": "boolean"
          }
        }
      }
    }
  }
}
```

Field Description

Field	Description
meta	Metadata information about the target server.
• serverDate	Format is (ISO8601): YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• odHost	IP or hostname.
• odVersion	If the client sent a correlationId in the request, it would be shown here in the response.
• correlationId	If the client sent a correlationId in the request, it would be shown here in the response.
timers	Timer information.
• timerId	Timer Identifier.
• userTimerName	For user timers: timer name defined by the user per Show. For static timers: this has the same value as “timerId”.
• type	Timer type (static, user).
• duration	Source total duration (if set) in milliseconds.
• startTime	Start time for the current timer. Format is (ISO8601): YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• countThrough	Allow count through.
• direction	Counting direction: “UP” or “DOWN” (case insensitive).
• running	True if this timer is running.
• currentValue	The current value for this timer. Calculated using: • (Direction-up): serverDate-startTime. • (Direction-down): duration-(severDate-startTime).
• clipName	Current clip name (when timer is a Clip Elapsed or Remained type).
• position	Current elapsed time in milliseconds.
• manualBehaviour	True if the timer has a manual behaviour,

Example

REST: GET http://<od>/server/floordirector/api/v1/timers?


```
{
  "meta": {
    "serverDate": "2009-04-11T14:22:07,125-0500",
    "odHost": "serverhost01",
    "odVersion": "16.3.0"
    "correlationId": "1000001"
  },
  "timers": [
    {
      "timerId": "Shot Time Elapsed",
      "userTimerName": "Shot Time Elapsed",
      "type": "static",
      "countThrough": true,
      "direction": "down",
      "running": true,
      "duration": 1000,
      "startTime": "2009-04-11T14:22:07,125-0500",
      "currentValue": "01:12",
      "clipName": "",
      "position": 7200,
      "manualBehaviour": false
    },
    {
      "timerId": "Timer01",
      "userTimerName": "BlackStormVO",
      "type": "user",
      "countThrough": false,
      "direction": "up",
      "running": true,
      "duration": 2000,
      "startTime": "2009-04-11T14:22:07,125-0500",
      "currentValue": "11:31"
      "clipName": "",
      "position": 7200,
      "manualBehaviour": false

    }
  ]
}
```

Shots

Gets shot information (OnAir/Prepared shots).

Parameters

URL Parameters	Description	Values	Required
type	The shots to be included in the response.	“onair” – only the onair shot. “prepared” – only the prepared shot. “”– empty for both onair and prepare shots.	No

Response: JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "meta": {
      "type": "object",
      "properties": {
        "serverDate": {
          "type": "string"
        },
        "odHost": {
          "type": "string"
        },
        "odVersion": {
          "type": "string"
        },
        "correlationId": {
          "type": "string"
        }
    "shots": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "type": {
            "type": "string"
          },
          "index": {
            "type": "string"
          },
          "slug": {
            "type": "string"
          },
          "shotName": {
            "type": "string"
          },
          "templateName": {
            "type": "string"
          },
          "transitionName": {
            "type": "string"
          }
        }
      }
    }
  }
}
```

Field Description

Field	Description
meta	Metadata information about the target server.
• serverDate	Format is (ISO8601): YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• odHost	IP or hostname.
• odVersion	If the client sent a correlationId in the request, it would be shown here in the response.
• correlationId	If the client sent a correlationId in the request, it would be shown here in the response.
shots	Shot information.
• type	Shot type (onair or prepared).
• slug	Slug.
• index	Shot index.
• shotName	Shot name.
• templateName	Shot template name.
• transitionName	Shot transition name.

Example

```
REST: GET http://<od>/server/floordirector/api/v1/shots?
```

```
{
  "meta": {
    "serverDate": "2009-04-11T14:22:07,125-0500",
    "odHost": "serverhost01",
    "odVersion": "16.3.0"
    "correlationId": "1000001"
  },
  "shots": [
    {
      "type": "onair",
      "index": "A1",
      "slug": "BREAKING NEWS",
      "shotName": "Shot CAM 101",
      "templateName": "CAM 101",
      "transitionName": "CUT"
    },
    {
      "type": "prepared",
      "index": "A2",
      "slug": "BREAK",
      "shotName": "Shot BLACK",
      "templateName": "BLACK",
      "transitionName": "WIPE"
    }
  ]
}
```

Info

Gets system information (Playing Rundown Name, controlling RC client, active and backup servers).

Parameters

URL Parameters	Description	Values	Required
None			

Response: JSON Schema


```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "meta": {
      "type": "object",
      "properties": {
        "serverDate": {
          "type": "string"
        },
        "odHost": {
          "type": "string"
        },
        "odVersion": {
          "type": "string"
        },
        "correlationId": {
          "type": "string"
        }
      }
    },
    "info": {
      "type": "object",
      "properties": {
        "playingRundownName": {
          "type": "string"
        },
        "controllingRcClient": {
          "type": "string"
        },
        "activeServer": {
          "type": "string"
        },
        "backupServer": {
          "type": "string"
        },
        "onAirModeEnabled": {
          "type": "boolean"
        },
        "devices": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "type": {
                "type": "string"
              },
              "identifier": {
                "type": "string"
              },
              "mode": {
                "type": "string"
              },
              "location": {
                "type": "string"
```

```
                },
                "status": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Field Description

Field	Description
meta	Metadata information about the target server.
• serverDate	Format is (ISO8601): YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• odHost	IP or hostname.
• odVersion	If the client sent a correlationId in the request, it would be shown here in the response.
• correlationId	If the client sent a correlationId in the request, it would be shown here in the response.
info	
• playingRundownName	Name of the playing rundown.
• controllingRcClient	Controlling RundownControl client.
• activeServer	Active Overdrive server hostname or IP address.
• backupServer	Backup Overdrive server hostname or IP address.
• onAirModeEnabled	True when the Playing RundownClient has the OnAir Mode enabled. False otherwise.
• devices	Device status information.
○ type	Type of OverDrive device or component.
○ identifier	Name or level of the current device or component.
○ mode	Current OverDrive redundant system mode of the device or component.
○ location	IP address or hostname or virtual location of the device or component.
○ status	Current status of the component.

Example

REST: GET http://<od>/server/floordirector/api/v1/info?


```
{
  "meta": {
    "serverDate": "2020-04-11T14:22:07,125-0500",
    "odHost": "serverhost01",
    "odVersion": "20.0.1"
    "correlationId": "1000001"
  },
  "info": {
    "playingRundownName": "[NRCS] INEWS Morning",
    "controllingRcClient": "192.168.10.24",
    "activeServer": "srvotto0001",
    "backupServer": "srvotto0002",
    "onAirModeEnabled": true,
    "devices":[
        {
          "type":"quickturn",
          "identifier":"QT-MOS-NAME",
          "mode":"primary"
          "location":"srvottcap001(chan1)",
          "status":"RECORDING"
        },
        {
          "type":"quickturn",
          "identifier":"QT-NON-MOS-NAME",
          "mode":"primary"
          "location":"srvottcap001(chan2)",
          "status":"STOP"
        }
    ]
  }
}
```

Cues

Gets the last ‘cue’ messages.

Parameters

URL Parameters	Description	Values	Required
None			

Response: JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "meta": {
      "type": "object",
      "properties": {
        "serverDate": {
          "type": "string"
        },
        "odHost": {
          "type": "string"
        },
        "odVersion": {
          "type": "string"
        },
        "correlationId": {
          "type": "string"
        }
      }
    },
    "cues": {
      "type": "object",
      "properties": {
        "lastOnAirMsg": {
          "type": "string"
        },
        "lastOnAirMsgDate": {
          "type": "string"
        },
        "lastPreparedMsg": {
          "type": "string"
        },
        "lastPreparedMsgDate": {
          "type": "string"
        }
      }
    }
  }
}
```

Field Description

Field	Description
meta	Metadata information about the target server.
• serverDate	Format is (ISO8601): YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• odHost	IP or hostname.
• odVersion	If the client sent a correlationId in the request, it would be shown here in the response.
• correlationId	If the client sent a correlationId in the request, it would be shown here in the response.
cues	
• lastOnAirMsg	The last on-air message.
• lastOnAirMsgDate	Date and time of the last on-air message in the format: YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• lastPreparedMsg	The last prepared message.
• lastPreparedMsgDate	Date and time of the last prepared message in the format: YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].

Example

```
REST: GET http://<od>/server/floordirector/api/v1/cues?
```

```
{
  "meta": {
    "serverDate": "2009-04-11T14:22:07,125-0500",
    "odHost": "serverhost01",
    "odVersion": "16.3.0"
    "correlationId": "1000001"
  },
  "cues": {
    "lastOnAirMsg": "Prepare CAM 1!",
    "lastOnAirMsgDate": "2009-04-11T14:22:07,125-0500",
    "lastPreparedMsg": "Prepare CAM 2!",
    "lastPreparedMsgDate": "2009-04-11T14:22:07,125-0500"
  }
}
```

Variable

Gets rundown variable view information (audio variable status).

Parameters

URL Parameters	Description	Values	Required
name	Only gets the variable status source with this name. If empty, returns the status all variables.	“variable name” – only the named variable. “”– empty for the statuses of all audio variables.	No

Response: JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-06/schema#",
  "type": "object",
  "properties": {
    "meta": {
      "type": "object",
      "properties": {
        "serverDate": {
          "type": "string"
        },
        "odHost": {
          "type": "string"
        },
        "odVersion": {
          "type": "string"
        }
      }
    },
    "statuses": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "variable": {
            "type": "object",
            "properties": {
              "variableId": {
                "type": "integer"
              },
              "variableName": {
                "type": "string"
              },
              "defaultSource": {
                "type": "string"
              }
            }
          },
          "next": {
            "type": "object",
            "properties": {
              "source": {
                "type": "string"
              },
              "status": {
                "type": "string"
              }
            }
          },
          "assigned": {
            "type": "object",
            "properties": {
              "source": {
                "type": "string"
```

```
              },
              "status": {
                "type": "string"
              }
            }
          },
          "nextShow": {
            "type": "object",
            "properties": {
              "source": {
                "type": "string"
              },
              "status": {
                "type": "string"
              }
            }
          },
          "assignedShow": {
            "type": "object",
            "properties": {
              "source": {
                "type": "string"
              },
              "status": {
                "type": "string"
              }
            }
          },
          "isLocked": {
            "type": "boolean"
          }
        }
      }
    }
  }
}
```


Field Description

Field	Description
meta	Metadata information about the target server.
• server-date	Format is YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• od-host	IP address or hostname.
• od-version	Overdrive version (MAYOR).(MINOR).
statuses	Audio variable view status information.
• variable	Audio variable information.
○ variableId	Variable identifier.
○ variableName	Variable name.
○ defaultSource	Default audio source.
• next	Next column information.
○ source	Next source to be applied.
○ status	“warning” when the source is different from the one that was set by a Variable Preset.
• assigned	Assigned column information.
○ source	Current Source applied.

○ status	“warning” when the source is different from the one that was set by a Variable Preset.
• nextShow	Next show column information.
○ source	Next Show Source to be applied.
○ status	“warning” when the source is different from the one that was set by a Variable Preset.
• assignedShow	Assigned show column information.
○ source	Current show source applied.
○ status	Returns “warning” when the source is different from the one that was set by a Variable Preset.
• isLocked	True when the variable has been locked from modifications, false otherwise.

Example

```
GET http://<od>/server/floordirector/v1/audio/variables?name=VARIABLE221
```


```
{
  "meta": {
    "server-date": "2009-04-11T14:22:07,125-05:00",
    "od-host": "srvottoddrv01",
    "od-version": "18.4"
  },
  "statuses": [
    {
      "variable": {
        "variableId": 157,
        "variableName": "VARIABLE221",
        "defaultSource": "Channel 121"
      },
      "next": {
        "source": "Default",
        "status": ""
      },

      "assigned": {
        "source": "Default",
        "status": "warning"
      },
      "nextShow": {
        "source": "None",
        "status": ""
      },
      "assignedShow": {
        "source": "None",
        "status": ""
      },
      "isLocked": false
    }
  ]
}
```

Subscription

Socket Only: It subscribes/unsubscribes the current client connection for receiving autonomous timing updates. The default behavior for new connections is being subscribed.

Parameters

URL Parameters	Description	Values	Required
type	Type of the request.	**optIn** – subscribe this connection. **optOut** – unsubscribe this connection.	Yes

Response: JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "meta": {
      "type": "object",
      "properties": {
        "serverDate": {
          "type": "string"
        },
        "odHost": {
          "type": "string"
        },
        "odVersion": {
          "type": "string"
        },
        "correlationId": {
          "type": "string"
        }
      }
    },
    "subscription": {
      "type": "object",
      "properties": {
        "response": {
          "type": "string"
        }
      }
    }
  }
}
```

Field Description

Field	Description
meta	Metadata information about the target server.
• serverDate	Format is (ISO8601): YYYY-MM-DD'T'hh:mm:ss[,ddd]['Z'].
• odHost	IP or hostname.
• odVersion	If the client sent a correlationId in the request, it would be shown here in the response.
• correlationId	If the client sent a correlationId in the request, it would be shown here in the response.
subscription	
• response	Message describing the response.

Example

```
Socket Request:
{
  "protocolVersion": "1",
  "endpoint": "subscription",
  "reqData": "type=optOut",
  "correlationId": "1000001"
}
```

```
Response:
{
  "meta": {
    "serverDate": "2009-04-11T14:22:07,125-0500",
    "odHost": "serverhost01",
    "odVersion": "16.3.0"
    "correlationId": "1000001"
  },
  "subscription": {
    "response": "Connection has successfully been removed from the subscriber list."
  }
}
```

Error Response

In the case of an error is thrown, the server will return an error message.

Response: JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "error": {
      "type": "string"
    },
    "errorCode": {
      "type": "integer"
    },
    "errorDescription": {
      "type": "string"
    }
  }
}
```

Field Description

Field	Description
error	Error name.
errorCode	Error code (internal).
errorDescription	Detailed description.

Example

```
{
  "error": "Bad Request",
  "errorCode": 400,
  "errorDescription": "Malformed request syntax!"
}
```

Field Codes

Code	Description
400	Bad Request, the server cannot or will not process the request due to an apparent client error (malformed message, invalid protocol version, missing parameters).
404	Endpoint Not Found, the client requested an invalid endpoint name.
500	Unknown Error.
503	Service Unavailable.