grass valley
A BELDEN BRAND

DENSITÉ/GECKOFLEX RESTFUL API

Reference Guide

M3059-0204-100

2017-03-08

www.grassvalley.com

Copyright and Trademark Notice

Terms and Conditions

Please read the following terms and conditions carefully. By using iControl Application Server documentation, you agree to the following terms and conditions.

Grass Valley hereby grants permission and license to owners of iControl Application Servers to use their product manuals for their own internal business use. Manuals for Grass Valley products may not be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose unless specifically authorized in writing by Grass Valley.

A Grass Valley manual may have been revised to reflect changes made to the product during its manufacturing life. Thus, different versions of a manual may exist for any given product. Care should be taken to ensure that one obtains the proper manual version for a specific product serial number.

Information in this document is subject to change without notice and does not represent a commitment on the part of Grass Valley.

Warranty information is available from the Legal Terms and Conditions section of Grass Valley's website (www.grassvalley.com).

Title	Densité/GeckoFlex RESTful API Reference Guide
Part Number	M3059-0204-100
Revision	2017-03-08, 20:03

Densité/GeckoFlex RESTful API

This document describes the Densité/GeckoFlex RESTful service. This service requires the *iControl Services Gateway* license (order code: IC-GATEWAY).

Introduction

As of iControl version 7.20, Densité Manager and GeckoFlex Manager provide a RESTful service to monitor and control cards housed in Densité, GV Node, and GeckoFlex frames.

The REST service is activated with the following configuration:

- Available via HTTP on the iControl Application Server
- Registered on port:
  - 5955 for Densité Manager 1
  - 5953 for Densité Manager 2
  - 5951 for Densité Manager 3
  - 5949 for Geckoflex Manager
- With base path:
  - `http://10.0.3.6:5955/densite` for Densité manager 1
  - `http://10.0.3.6:5953/densite` for Densité manager 2
  - `http://10.0.3.6:5951/densite` for Densité manager 3
  - `http://10.0.3.6:5949/geckoflex` for Geckoflex manager

A WADL file describing the service is available in the base path for each registered manager:

- `http://10.0.3.6:5955/densite/application.wadl` for Densité manager 1
- `http://10.0.3.6:5953/densite/application.wadl` for Densité manager 2
- `http://10.0.3.6:5951/densite/application.wadl` for Densité manager 3
- `http://10.0.3.6:5949/geckoflex/application.wadl` for Geckoflex manager

In this document:

- The IP address `10.0.3.6` represents the iControl Application Server.
- URLs refer to *Densité Manager 1* (port 5955). For another Densité manager or Geckoflex manager, use the appropriate base path and port number.

Service description

Individual requests

Ping

This method is useful to ensure that the REST service is running and accessible at the appropriate path.

Path	`http://10.0.3.6:5955/densite/ping`
Action	GET
Input	None
Input format	N/A
Output	pong
Output media type	text/plain

HTTP request example

```
GET /densite/ping HTTP/1.1
Host: 10.0.3.6:5955
Cache-Control: no-cache
```

HTTP response example

```
pong
```

Get cards

This method is used to obtain the list of all cards that are accessible.

Path	`http://10.0.3.6:5955/densite/cards`
Action	GET
Headers	Authorization: Basic
Input	None
Input format	N/A
Output	CARD_LIST_JSON
Output media type	application/json

HTTP request example

```
GET /densite/cards HTTP/1.1
Host: 10.0.3.6:5955
Authorization: Basic
Cache-Control: no-cache
```

HTTP response example

{"cards":[{"id":"AppServer_GV-Node_Densite_SLOT_17_177","name":"IFM-2T","frameId":"AppServer_GV-Node","slot":17,"devId":177,"version":"1.0.0"}]}

Get all parameter names

This method is used to obtain all parameter names on a specific card.

Path	`http://10.0.3.6:5955/densite/cards/{cardId}/parameters`
Action	GET
Headers	Authorization: Basic
Input	cardId: The ID of the card to query
Input format	In the path
Output	PARAMETER_JSON
Output media type	application/json

HTTP request example

GET /densite/cards/MC_20_Densite_SLOT_2_64/parameters/

Host: 10.0.3.6:5955

Authorization: Basic

Cache-Control: no-cache

HTTP response example

{"cardId":"MC_20_Densite_SLOT_2_64","parameters":{"ids":["aMuteAES6","aMuteAES5","aMuteAES8","aMuteAES7","aMuteAES3"]}}

Get one parameter

This method is used to obtain the current value of a specific parameter on a specific card.

Path	`http://10.0.3.6:5955/densite/cards/{cardId}/parameters/{parameterId}`
Action	GET
Headers	Authorization: Basic
Input	• cardId: The ID of the card to query • parameterId: The name of the parameter to get
Input format	In the path
Output	PARAMETER_JSON
Output media type	application/json

HTTP request example

GET /densite/cards/AppServer_GV-Node_Densite_SLOT_17_177/parameters/PARAM::IFM.EthRtpIn[132].srcIpAddr

HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

Cache-Control: no-cache

HTTP response example

{"parameterId":"PARAM::IFM.EthRtpIn[132].srcIpAddr""value":[192,168,0,7]}

Get parameter details

This method is used to obtain detailed information for a specific parameter on a specific card.

Path	`http://10.0.3.6:5955/densite/cards/{cardId}/parameters/{parameterId}/info`
Action	GET
Headers	Authorization: Basic
Input	• cardId: The ID of the card to query • parameterId: The name of the parameter to get
Input format	• cardId: In the path • parameterId: In the path
Output	parameterInfoDto
Output media type	application/json

HTTP request example

GET /densite/cards/MC33_MyFrame_Densite_SLOT_17_114/parameters/dLipsyncAud6StatusGRP/info

Host: 10.0.3.6:5955

Authorization: Basic

Content-Type: application/json

Cache-Control: no-cache

HTTP response example

{"choices":[{"index":"OFF","message":"OFF","active":true},{"index":"A","message":"A","active":true},
{"index":"B","message":"B","active":true}],"valueType":"String","parameterId":"dLipsyncAud6StatusGRP","name":"Group","active":true,"type":"choice","accessType":"Write only"}

Get many parameters

This method is used to obtain the current value of many parameters on a specific card.

Path	`http://10.0.3.6:5955/densite/cards/{cardId}/parameters/read`
Action	POST
Headers	• Authorization: Basic • Content-Type: application/json
Input	• cardId: The ID of the card to query • Content: LIST_OF_ID_JSON
Input format	• cardId: In the path • Content: JSON
Output	LIST_OF_PARAMETER_JSON
Output media type	application/json

HTTP request example

POST /densite/cards/AppServer_GV-Node_Densite_SLOT_17_177/parameters/read HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

Content-Type: application/json

Cache-Control: no-cache

{"ids":["PARAM::IFM.EthRtpOut[25].destIpAddr",
"PARAM::IFM.EthRtpOut[25].destIpPort"]}

HTTP response example

{"parameters":[{"parameterId":"PARAM::IFM.EthRtpOut[25].destIpPort","value":0},{"parameterId":"PARAM::IFM.EthRtpOut[25].destIpAddr","value":[0,0,0,0]}]}

Set many parameters

This method is used to set the value of many parameters on a specific card.

Path	`http://10.0.3.6:5955/densite/cards/{cardId}/parameters/write`
Action	POST
Headers	• Authorization: Basic • Content-Type: application/json
Input	• cardId: The ID of the card to query • Content: LIST_OF_PARAMETER_JSON
Input format	• cardId: In the path • Content: JSON
Output	HTTP status 204 (no content)
Output media type	N/A

HTTP request example

POST /densite/cards/AppServer_GV-Node_Densite_SLOT_17_177/parameters/write HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

Content-Type: application/json

Cache-Control: no-cache

{"parameters":[{"parameterId":"PARAM::IFM.EthRtpOut[25].destIpPort","value":0},{"parameterId":"PARAM::IFM.EthRtpOut[25].destIpAddr","value":[0,0,0,0]}]}

HTTP response example

204: no content

Session based requests

Sessions allow receiving notifications of changed parameters on specific cards. The content of the session output buffer is retrieved by a long-polling method. If the buffer contains data, this method returns immediately with the data. If not, then the call is held until data is acquired or a delay of 30 seconds is elapsed.

Connect

This method is used to create a new session, and retrieve its ID. After creation, the session output buffer will contain the hardware information (available frames and cards).

Note: A session times out if the output buffer has not been accessed for 40 seconds.

Path	`http://10.0.3.6:5955/densite/sessions/connect`
Action	POST
Headers	Authorization: Basic
Input	None
Input format	N/A
Output	SESSION_ID_JSON
Output media type	application/json

HTTP request example

POST /densite/sessions/connect HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

HTTP response example

{"id":"9b4ae4f9-9fe7-4a07-a0b8-5afc93b4c171"}

Disconnect

This method is used to terminate a session.

> **Note:** A session times out if the output buffer has not been accessed for 40 seconds.

Path	`http://10.0.3.6:5955/densite/sessions/{sessionId}`
Action	DELETE
Headers	Authorization: Basic
Input	sessionId: The ID of the session
Input format	In the path
Output	HTTP status 200 (OK)
Output media type	N/A

HTTP request example

DELETE /densite/sessions/9b4ae4f9-9fe7-4a07-a0b8-5afc93b4c171 HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

HTTP response example

Success 200: OK.

Session closed

Get session data

This method is used to retrieve the session output buffer if data is waiting, or is held until data is available or 30 seconds.

Path	`http://10.0.3.6:5955/densite/sessions/{sessionId}`
Action	GET
Headers	Authorization: Basic
Input	sessionId: The ID of the session
Input format	In the path
Output	• HTTP status 304 (not modified) • SESSION_DATA_JSON
Output media type	• Status: N/A • Data: application/json

HTTP request example

GET /densite/sessions/9b4ae4f9-9fe7-4a07-a0b8-5afc93b4c171 HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

HTTP response example

{"hardware": [
{"cardContainer": {"id": "Appserver_Gv-Node_Densite", "name": "Gv-Node", "ipAddress": "10.0.3.7" }, "hardwareType": "card container", "action": "added"},
{"card": {"id": "Appserver_Gv-Node_Densite_SLOT_18_65530", "name": "ETH3-REF", "frameId": "Appserver_Gv-Node_Densite", "slot": 18, "devId": 65530,"version": "1.0.2"}, "hardwareType": "card", "action": "added"},
{"card": {"id": "Appserver_Gv-Node_Densite_SLOT_17_177", "name": "IFM-2T", "frameId": "Appserver_Gv-Node_Densite", "slot": 17, "devId": 177, "version": "1.0.0"}, "hardwareType": "card", "action": "added"}
]}

Register card for notification

This method is used to register a card, and be notified when one of its parameters changes. After registration, the session output buffer will contain all the current information about the card (list of parameters, parameter information, parameter values).

Path	`http://10.0.3.6:5955/densite/sessions/{sessionId}/cards/{cardId}`
Action	PUT
Headers	Authorization: Basic
Input	• sessionId: The ID of the session • cardId: The ID of the card to register
Input format	In the path
Output	HTTP status 204 (no content)
Output media type	N/A

HTTP request example

PUT /densite/sessions/9b4ae4f9-9fe7-4a07-a0b8-5afc93b4c171/cards/Appserver_Gv-Node_Densite_SLOT_17_177 HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

HTTP response example

204: No Content

Unregister card from notification

This method is used to unregister a card from sending notifications.

Path	`http://10.0.3.6:5955/densite/sessions/{sessionId}/cards/{cardId}`
Action	DELETE

Headers	Authorization: Basic
Input	• sessionId: The ID of the session • cardId: The ID of the card to register
Input format	In the path
Output	HTTP status 204 (no content)
Output media type	N/A

HTTP request example

DELETE /densite/sessions/9b4ae4f9-9fe7-4a07-a0b8-5afc93b4c171/cards/Appserver_Gv-Node_Densite_SLOT_17_177 HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

HTTP response example

204: no content

Get one parameter

This method is used to request that the value of a parameter on a specific card be added to the session output buffer. This method differs from the individual request Get one parameter, in that the result is not returned synchronously but *asynchronously* via the session output buffer.

Path	`http://10.0.3.6:5955/densite/sessions/{sessionId}/cards/{cardId}/parameters/{parameterId}`
Action	GET
Headers	Authorization: Basic
Input	• sessionId: The ID of the session • cardId: The ID of the card to register • parameterId: The name of the parameter to get
Input format	In the path
Output	HTTP status 204 (no content)
Output media type	N/A

HTTP request example

GET /densite/sessions/9b4ae4f9-9fe7-4a07-a0b8-5afc93b4c171/cards/AppServer_GV-Node_Densite_SLOT_17_177/parameters/PARAM::IFM.EthRtpIn[132].srcIpAddr HTTP/1.1

Host: 10.0.3.6:5955

Authorization: Basic

HTTP response example

204: no content

JSON objects

CARD_LIST_JSON

```
{"cards":[CARD_JSON, ..., CARD_JSON]}
```

- cards: An array of CARD_JSON.

CARD_JSON

```
{"id": "AppServer_GV-Node_Densite_SLOT_2_168","name": "IPG-
3901","frameId": "AppServer_GV-Node","slot": 12,"devId": 168,"version":
"1.0.0"}
```

- id: The ID of the card. It is used to reference this card in the REST service.
- name: The name of the card (e.g., IPG-3901, MAP-3901, IFM-2T).
- frameId: The ID of the frame that contains the card.
- slot: The number of the slot in which the card is located.
- devId: The device ID that uniquely identify the card type.
- version: The version of the firmware present on the card.

PARAMETER_JSON

```
{"parameterId": "PARAM::PRESET.card.user[0].name","value": "CUSTOM"}
```

- parameterId: The name of the requested parameter.
- value: The value of the parameter. This value can be a string, a boolean, a number, or an array.

parameterInfoDto

```
{"choices":[{"index":"OFF","message":"OFF","active":true},{"index":"ON","
message":"ON","active":true}],"valueType":"String","parameterId":"aMuteAE
S6","name":"Mute","active":true,"type":"choice","accessType":"Read
write"}
```

- choices: An array of the choices available from the card's user interface.
- valueType: The type of the values.
- parameterId: The name of the parameter (i.e., its key).
- name: The text label that appears on the card's user interface for the parameter.
- active: The current status of the parameter (i.e., active or not active).
- type: The type of the parameter.
- accesType: Read or write access.

LIST_OF_ID_JSON

```
{"ids":["PARAM::PRESET.card.user[0].name", ...,
"PARAM::PRESET.card.user[1].name"]}
```

- ids: An array of the names of the parameters to retrieve.

LIST_OF_PARAMETER_JSON

```
{"parameters":[PARAMETER_JSON, ..., PARAMETER_JSON]}
```

SESSION_ID_JSON

```
{"id":"9b4ae4f9-9fe7-4a07-a0b8-5afc93b4c171"}
```

SESSION_DATA_JSON

```
{"hardware":[SESSION_DATA_HARDWARE_JSON, ...,
SESSION_DATA_HARDWARE_JSON],
 "values":[SESSION_DATA_VALUES_JSON, ..., SESSION_DATA_VALUES_JSON],
 "infos":[SESSION_DATA_INFOS_JSON, ..., SESSION_DATA_INFOS_JSON],
 "parameters":[SESSION_DATA_PARAMETER_LIST_JSON, ...,
SESSION_DATA_PARAMETER_LIST_JSON]}
```

SESSION_DATA_HARDWARE_JSON

SESSION_DATA_HARDWARE_CARD_CONTAINER_JSON or
SESSION_DATA_HARDWARE_CARD_JSON

SESSION_DATA_HARDWARE_CARD_CONTAINER_JSON

```
{"cardContainer": {"id": "Appserver_Gv-Node_Densite", "name": "GvNode",
"ipAddress": "10.0.3.7"}, "hardwareType": "card container","action":
"added"}
```

- id: The ID that represents the frame. It is used to reference this frame in the REST service.
- name: The name of the frame.
- ipAddress: The IP address of the frame.
- hardwareType: Will be “card container” if the data represents a frame.
- action: Will be “added” for a frame that is now available, or “removed” for a frame that is not available anymore.

SESSION_DATA_HARDWARE_CARD_JSON

```
{"card": {"id": "Appserver_Gv-Node_Densite_SLOT_15_161", "name": "MAP-
3901", "frameId": "Appserver_Gv-Node_Densite", "slot": 15, "devId": 161,
"version": "1.0.0"}, "hardwareType": "card", "action": "added"}
```

- id: The ID that represents the card.
- name: The name of the card (e.g., IPG-3901, MAP-3901, IFM-2T).
- frameId: The ID of the frame that contains the card.
- slot: The number of the slot in which the card is located.
- devId: The device ID that uniquely identify the card type.
- version: The version of the firmware present on the card.
- hardwareType: Will be “card” if the data represents a card.

- action: Will be “added” for a card that is now available, or “removed” for a frame that is not available anymore.

SESSION_DATA_VALUES_JSON

```
{"cardId":"Appserver_Gv-Node_Densite_SLOT_15_161",
"parameter":PARAMETER_JSON}
```

cardId: The ID of the card.

parameter: The JSON object that contains the parameterId and value.

SESSION_DATA_INFOS_JSON

```
{"cardId":"Appserver_Gv-Node_Densite_SLOT_15_161",
"info":PARAMETER_INFO_JSON}
```

- cardId: The ID of the card.
- info: The JSON object that contains the description of the parameter.

SESSION_DATA_PARAMETER_LIST_JSON

```
{"cardId":"Appserver_Gv-Node_Densite_SLOT_15_161",
"parameters":{"ids":["PARAM::IFM.EthRtpIn[0].srcIpPort", ...,
"PARAM::IFM.EthRtpIn[143].srcIpPort"]}}
```

- cardId: The ID of the card.
- ids: An array that contains the name of all the parameters exposed by the card.

Supported Cards

The Densité/GeckoFlex RESTful API supports all cards housed in Densité, GV Node, and GeckoFlex frames.

Grass Valley Technical Support

For technical assistance, contact our international support center, at 1-800-547-8949 (US and Canada) or +1 530 478 4148.

To obtain a local phone number for the support center nearest you, please consult the Contact Us section of Grass Valley's website (www.grassvalley.com).

An online form for e-mail contact is also available from the website.

Corporate Head Office

Grass Valley
3499 Douglas-B.-Floreani
St-Laurent, Quebec H4S 2C6
Canada
Telephone: +1 514 333 1772
Fax: +1 514 333 9828
www.grassvalley.com