DynaColor Standard API - Interface Specification

Revision: 1.03

Date: 2007-Dec-28

TABLE OF CONTENTS

- DOCUMENT HISTORY
- 1 OVERVIEW
  - 1.1 Product and firmware versions
- 2 REFERENCES
- 3 DEFINITIONS
  - 3.1 General notations
    - 3.1.1 General abbreviations
    - 3.1.2 Style convention
    - 3.1.3 General CGI URL syntax and parameters
    - 3.1.4 Parameter value convention
- 4 INTERFACE SPECIFICATION
  - 4.1 Server responses
    - 4.1.1 HTTP status codes
- 5 API GROUPS
  - 5.1 General
    - 5.1.1 Update and list parameters and their values
      - 5.1.1.1 List parameters
      - 5.1.1.2 List output format
      - 5.1.1.3 Update parameters
    - 5.1.2 Add, modify and delete users
    - 5.1.3 List user information
    - 5.1.4 Get, modify snapshot path
    - 5.1.5 Iris adjustment
    - 5.1.6 Factory default
    - 5.1.7 Hard factory default
    - 5.1.8 Firmware upgrade
    - 5.1.9 Restart server
    - 5.1.10 Server report
    - 5.1.11 System logs
    - 5.1.12 System date and time
      - 5.1.12.1 Get system date and time
      - 5.1.12.2 Set system date and time

- 5.2 Video
  - 5.2.1 Setup
  - 5.2.2 Connect/Disconnect video stream
- 5.3 Audio
  - 5.3.1 Audio control

DOCUMENT HISTORY

Version	Date	Comment
1.00	2007-Oct-9	Initial version.
1.01	2007-Oct-16	Add firmware upgrade
1.02	2007-Oct-29	Add snapshot function
1.03	2007-Dec-28	Add iris adjustment

1 OVERVIEW

This document specifies the external HTTP-based application programming interface of the DynaColor megapixel camera with firmware version z20070927 and above.

The HTTP-based video interface provides the functionality for requesting images and for getting and setting internal parameter values. The image and CGI-requests are handled by the built-in Web server in the camera.

1.1 Product and firmware versions

The support for the HTTP API is product and firmware dependent. Please refer to the Release Notes for the actual product for compliance information.

2 REFERENCES

HTTP protocol

- Hypertext Transfer Protocol -- HTTP/1.0

External application programming interfaces (Client side)

- DynaColor API parameters

3 DEFINITIONS

This section contains information on general usage of this document.

3.1 General notation

3.1.1 General abbreviations

The following abbreviations are used throughout this document

CGI	*Common Gateway Interface* - a standardized method of communication between a client (e.g. a web browser) and a server (e.g. a web server).
N/A	*Not applicable* - a feature/parameter/value is of no use in a specific task
URL	RFC 1738 describes the syntax and semantics for a compact string representation for a resource available via the Internet. These strings are called "Uniform Resource Locators" (URLs).
URI	A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource. RFC 2396 describes the generic syntax of URI.

3.1.2 Style convention

In URL syntax and in descriptions of CGI parameters, text in italics within angle brackets denotes content that should be replaced with either a value or a string. When replacing the text string, the angle brackets must also be replaced. An example of this is the description of the name for the server, denoted with <*servername*> in the URL syntax description below, which is replaced with the string `myserver` in the URL syntax example, also shown below.

URL syntax is written with the word "Syntax:" shown in bold face, followed by a box with the referred syntax, as shown below. The name of the server is written as <*servername*>. This is intended to be replaced with the name of the actual server. This can either be a name, e.g. "thecam" or "thecam.adomain.net" or the associated IP number for the server, e.g. 192.168.0.250.

Syntax:

```
http://<servername>/cgi-bin/admin/userinfo.cgi
```

A description of returned data is written with "Return:" in bold face, followed by the returned data in a box. All data returned as HTTP-formatted, i.e. starting with the string HTTP, is line-separated with a Carriage Return and Line Feed (CRLF) printed as \r\n.

Return:

```
HTTP/1.0 <HTTP code> <HTTP text>\r\n
```

URL syntax examples are written with "Example:" in bold face, followed by a short description and a light grey box with the example.

Example: Request user privacy.

```
http://myserver/cgi-bin/admin/privacy.cgi
```

Examples of what can be returned by the server from a request are written with "Example:" in bold face, followed by a short description and a light grey box with an example of the returned data.

Example: Returned data after a successful request.

```
HTTP/1.0 200 Ok\r\n
```

3.1.3 General CGI URL syntax and parameters

CGI URLs are written in lower-case. CGI parameters are written in lower-case and as one word. When the CGI request includes internal camera parameters, the internal parameters must be written exactly as named in the camera or video server. For the POST method, the parameters must be included in the body of the HTTP request. The CGIs are organized in function related directories under the *cgi-bin* directory. The file extension of the CGI is required.

Syntax:

```
http://<servername>/cgi-bin/<subdir>[/<subdir>...]/<cgi>.<ext>
[?<parameter>=<value>[&<parameter>=<value>...]]
```

Example: List the Network parameters.

```
http://<servername>/cgi-bin/admin/param.cgi?action=list&group=Network
```

3.1.4 Parameter value convention

In tables defining CGI parameters and supported parameter values, the default value for optional parameters is system configured.

4 INTERFACE SPECIFICATION

4.1 Server responses

4.1.1 HTTP status codes

The built-in Web server uses the standard HTTP status codes.

Return:

```
HTTP/1.0 <HTTP code> <HTTP text>\r\n
```

with the following HTTP code and meanings

HTTP code	HTTP text	Description
200	OK	The request has succeeded, but an application error can still occur, which will be returned as an application error code.
204	No Content	The server has fulfilled the request, but there is no new information to send back.
302	Moved Temporarily	The server redirects the request to the URI given in the Location header.
400	Bad Request	The request had bad syntax or was impossible to fulfill.
401	Unauthorized	The request requires user authentication or the authorization has been refused.
404	Not Found	The server has not found anything matching the request.
409	Conflict	The request could not be completed due to a conflict with the current state of the resource.
500	Internal Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
503	Service Unavailable	The server is unable to handle the request due to temporary overload.

Example: Request includes invalid file names.

```
HTTP/1.0 404 Not Found\r\n
```

5 API GROUPS

To make it easier for developers to get an idea of which API requests are supported for different products, the requests have been grouped together. Information about which groups are supported can be found in the product-specific release notes document, available for download from the Dynacolor web site.

5.1 General

The requests specified in the General section are supported by all video products with firmware version z20070921 and below.

5.1.1 Update and list parameters and their values

Note:

- The parameter is specified in the parameter document.
- The URL must follow the standard way of writing a URL, (RFC 2396: Uniform Resource Identifiers (URI) Generic Syntax); that is, spaces and other reserved characters (";", "/", "?", ":", "@", "&", "=", "+", "," and "$") within a <parameter> or a <value> must be replaced with %<ASCII hex>. For example, in the string My camera, the space will have to be replaced with %20, My%20camera.

Method: GET/POST

Syntax:

```
http://<servername>/cgi-bin/admin/param.cgi?
<parameter>=<value>[&<parameter>=<value>...]
```

with the following parameter and values

<parameter>=<value>	Values	Description
action=<*string*>	add, remove, update or list	Specifies the action to take. Depending on this parameter, various parameters may be set, as described in the following sections.

5.1.1.1 List parameters

Syntax:

```
http://<servername>/cgi-bin/admin/param.cgi?action=list
[&<parameter>=<value>...]
```

with the following parameter and values

<parameter>=<value>	Values	Description
group=<*string*>[,<*string*>...]	<*group*[.*name*]>[,<*group*[.*name*]>...]	Returns the value of the camera parameter named <*group*>.<*name*>. If <*name*> is omitted, all the parameters of the <*group*> are returned. The camera parameters must be entered exactly as they are named in the camera or video server. Wildcard (*) can be used when listing parameters. See example below. If this parameter is omitted, all parameters in the device are returned.

Example: List the Network parameters.

```
http://myserver/cgi-bin/admin/param.cgi?action=list&group=Network
```

Example: List the names of all Event parameters.

```
http://myserver/cgi-bin/admin/param.cgi?action=list&group=Event
```

5.1.1.2 List output format

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
<parameter pair>

where <parameter pair> is

<parameter>=<value>\n
```

```
[ <parameter pair> ]
```

Example: Network query response.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
root.Network.IPAddress=192.168.0.250\n
root.Network.SubnetMask=255.255.255.0\n
```

If the CGI request includes an invalid parameter value, the server returns an error message.

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
# Error: <description>\n
```

5.1.1.3 Update parameters

Syntax:

```
http://<servername>/cgi-bin/admin/param.cgi?action=update
[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	Values	Description
<*string*>=<*string*>	<*group.name*>=<*value*>	Assigns <*value*> to the parameter <*group.name*>. The <*value*> must be URL-encoded when it contains non-alphanumeric characters. The camera parameters must be entered exactly as named in the camera or the video server.

Example: Set the exposure mode to auto.

```
http://myserver/cgi-bin/operator/param.cgi?
action=update&ImageSource.I0.Sensor.Exposure=auto
```

Example: Set the event enable.

```
http://myserver/cgi-bin/operator/param.cgi?
action=update& Event.E0.Enabled=yes
```

5.1.2 Add, modify and delete users

Add a new user with password and group membership, modify the information and remove a user.

Note: This request requires root access (root authorization).

Method: GET/POST

Syntax:

```
http://<servername>/cgi-bin/admin/pwdgrp.cgi?
<parameter>=<value>[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	Values	Description
action=<*string*>	add, update, remove or get	add = create a new user account. update = change account information of specified parameters if the account exists. remove = remove an existing account if it exists. get = get a list of the users which belong to each group defined.
user=<*string*>	<*string*>	The user account name, a non-existing user name. Valid characters are a thru z, A thru Z and 0 thru 9.
pwd=<*string*>	<*string*>	The unencrypted password of the account. Valid characters are a thru z, A thru Z and 0 thru 9.

sgrp=<*string*>:[<*string*>...]	<*string*>[,<*string*>...]	Colon separated existing secondary group names of the account. Ex: dido：camctrl：talk：listen

Example: Create a new administrator account.

```
http://myserver/cgi-bin/admin/pwdgrp.cgi?action=add&user=joe&pwd=foo&sgrp=dido:camctrl:talk:listen
```

Example: Change the password of an existing account.

```
http://myserver/cgi-bin/admin/pwdgrp.cgi?action=update&user=joe&pwd=bar
```

Example: Remove an account.

```
http://myserver/cgi-bin/admin/pwdgrp.cgi?action=remove&user=joe
```

Example: List groups and users.

```
http://myserver/cgi-bin/admin/pwdgrp.cgi?action=get
```

5.1.3 List users information

List the user information with password or privacy.

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/privacy.cgi
```

Example: List the username and privacy

```
http://myserver/cgi-bin/admin/privacy.cgi?
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
```

```
Username:Dido:Camset:Talk:Listen \n
Admin:1:1:1:1\n
```

Syntax:

```
http://<servername>/cgi-bin/admin/userinfo.cgi
```

Example: List the username and password.

```
http://myserver/cgi-bin/admin/userinfo.cgi?
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
List username and password\n
Admin:1234\n
```

5.1.4 Get, modify snapshot path

Get or modify Admin snapshot path, Admin can capture images by the web page snapshot button, and the images are stored at the path you set.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/snapshot.cgi? <parameter>=<value>]
```

with the following parameters and values

<parameter>=<value>	Values	Description
path=<*string*>	<string>	Valid character: A-Za-z0-9and some special tokens _/\~!@#$%^&+-:
action=<string>	get set	Get the Admin snapshot path.

		Set the Admin snapshot path, with action=set parameter path is required.

Example: Set the Admin snapshot path to C:\capture

```
http://myserver/cgi-bin/admin/snapshot.cgi?action=set&path=C%3A%5Ccapture
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\n
\n
OK\n
```

5.1.5 Iris Adjustment

Adjust Iris when you use auto iris lens. With this function, system will automatically adjust with different lens.

Note: This requires administrator access (administrator authorization)

Before adjustment, you should check if the auto iris lens is set up first. Second, you should image a gray scale chart type I (Gamma = 1) over the entire screen.

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/irisadjust.cgi
```

5.1.6 Factory default

Reload factory default. All parameters except Network.BootProto, Network.IPAddress, Network.SubnetMask, Network.Broadcast, Network.DefaultRouter and Network port are set to their factory default values.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/factorydefault.cgi
```

5.1.7 Hard factory default

Reload factory default. All parameters are set to their factory default value.

Note: This request requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/hardfactorydefault.cgi
```

5.1.8 Firmware upgrade

Upgrade the firmware version.

Note: This requires administrator access (administrator authorization).

Method: POST

Syntax:

```
http://<servername>/cgi-bin/admin/firmwareupgrade.cgi[?<parameter>=<value>]
```

with the following parameters and values

<parameter>=<value>	Values	Description
filename=<*string*>	uImage, cameraFw, userland.jffs2	Specifies the filename of firmware upgrade. uImage= kernel package binary file. cameraFw = camera parameters binary file. userland.jffs2 = JFFS2 image binary file.

The file content is provided in the HTTP body according to the format given in RFC 1867. The body is created automatically by the browser if using HTML form with input type "file".

Example:

```
POST /cgi-bin/admin/firmwareupgrade.cgi?type=normal HTTP/1.0\r\n
Content-Type: multipart/form-data; boundary=AsCg5y\r\n
Content-Length: <content length>\r\n
\r\n
--AsCg5y\r\n
Content-Disposition: form-data; name="firmware.bin"; filename="firmware.bin"\r\n
Content-Type: application/octet-stream\r\n
\r\n
<firmware file content>
\r\n
--AsCg5y--\r\n
```

5.1.9 Restart server

Restart server.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/restart.cgi
```

5.1.10 Server report

This CGI request generates and returns a server report. This report is useful as an input when requesting support. The report includes product information, parameter settings and system logs.

Note: This requires administrator access (administrator authorization).

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/serverreport.cgi
```

5.1.11 System logs

Get system log information.

Note: This requires administrator access (administrator authorization).

Note: The response is product/release-dependent.

Method: GET

Syntax:

```
http://<servername>/cgi-bin/admin/systemlog.cgi
```

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
<system log information>
```

5.1.12 System date and time

Get or set the system date and time.

Method: GET/POST

Syntax:

```
http://<servername>/cgi-bin/admin/date.cgi?<parameter>=<value>
```

with the following parameter and values

<parameter>=<value>	Values	Description
Action=*<string>*	get or set	Specifies what to do. get = get the current date and time. set = set the current date and/or time.

5.1.12.1 Get system date and time

Syntax:

```
http://<servername>/cgi-bin/admin/date.cgi?action=get
```

Return:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
<month> <day>, <year> <hour>:<minute>:<second>\r\n
```

Example:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Apr 03, 2003 15:16:04\r\n
```

5.1.12.2 Set system date and time

Syntax:

```
http://<servername>/cgi-bin/admin/date.cgi?action=set[&<parameter>=<value>...]
```

with the following parameters and values

<parameter>=<value>	Values	Description
year=<*int*>	2007 - 2099	Current year.
Month=<*int*>	1 - 12	Current month.
day=<*int*>	1 - 31	Current day.
hour=<*int*>	0 - 23	Current hour.
minute=<*int*>	0 - 59	Current minute.
second=<*int*>	0 - 59	Current second.

The set action produces one of the following server responses:

Return: A successful *set.*

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
OK\r\n
```

Return: A failed *set*. Settings or syntax are probably incorrect.

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
Request failed: <error message>\r\n
```

Example: Set the date.

```
http://myserver/cgi-bin/admin/date.cgi?action=set&year=2005&month=4&day=3
```

Response:

```
HTTP/1.0 200 OK\r\n
Content-Type: text/plain\r\n
\r\n
OK\r\n
```

5.2 Video

5.2.1 Setup

Install the NSPlayer and ipp41_runtime_installer and create an object in the web page as below.

```
<html>

<head>
<title>Example page1</title>
</head>

<body>
<object id="NSPlayer1" width=640 height=480
 classid="CLSID:3C738339-D7C3-4044-868E-E3D5B0FC0F75">
    <param name="_Version" value="65536">
    <param name="_ExtentX" value ="2646">
    <param name="_ExtentY" value ="1323">
    <param name="_StockProps" value ="0">
</object>
</body>

</html>
```

5.2.2 Connect/Disconnect video stream

Connect/Disconnect a video stream by UDP or HTTP with default resolution and compression as defined in the system configuration.

Syntax: set to MPEG-4

```
NSPlayer1.VideoFormat = 0
```

Syntax: set to MJPEG

```
NSPlayer1.VideoFormat = 1
```

Syntax: Connect

```
NSPlayer1.Start( “rtsp://myserver/” )
NSPlayer1.Start( “http://myserver:8008/” )  (mjpeg only!)
```

Syntax: Disconnect

```
NSPlayer1.Stop()
```

Example:

```
<html>
<head>
<title>Example page2</title>
</head>
<body>
<center>
<object id="NSPlayer1" width=640 height=480
 classid="CLSID:3C738339-D7C3-4044-868E-E3D5B0FC0F75">
    <param name="_Version" value="65536">
    <param name="_ExtentX" value ="2646">
    <param name="_ExtentY" value ="1323">
    <param name="_StockProps" value ="0">
</object>
</center>
<center>

<form name="form">
<!-- Destination URL of connection -->
URL <input name="URL" type="TEXT" value="rtsp://192.168.0.250/" ><BR>
<!-- Video Format -->
VIDEO
<input name="Video" type="RADIO" onclick=RADIO_OnClick() value="MPEG4">MPEG4
<input name="Video" type="RADIO" onclick=RADIO_OnClick() value="MJPEG" checked>MJPEG<BR>
<!-- Connection -->
<input name = "Connect" type = "button" value = "Connect">
<!-- Disconnection -->
<input name = "Disconnect" type = "button" value = "Disconnect">

</center>
</form>
```

```
<script language = "VBScript">
<!--
Dim TheForm

Sub RADIO_OnClick   'Setting for Video Format
    Set TheForm = Document.form
    If TheForm.elements(1).checked = True Then
        NSPlayer1.VideoFormat = 1    'Set MPEG
    End if
    If TheForm.elements(2).checked = True Then
        NSPlayer1.VideoFormat = 0    'Set MJPEG
    End If
End Sub

Sub Connect_OnClick 'Connection
    Set TheForm = Document.form
    call NSPlayer1.Start(TheForm.URL.Value) 'Connection
End Sub

Sub Disconnect_OnClick  'Disconnection
    call NSPlayer1.Stop()   'Disconnection
End Sub
-->
</script>

</body>
</html>
```

5.3 Audio

5.3.1 Audio control

To set mute on/off and send data yes or not at audio function.

Syntax: mute on/off

```
NSPlayer1.Mute(TRUE)
NSPlayer1.Mute(FALSE)
```

Syntax: Send audio data

```
NSPlayer1.Mic(TRUE)
NSPlayer1.Mic(FALSE)
```

Example:

```
<html>
<head>
<title>Example page2</title>
</head>
<body>
<center>
<object id="NSPlayer1" width=640 height=480
 classid="CLSID:3C738339-D7C3-4044-868E-E3D5B0FC0F75">
    <param name="_Version" value="65536">
    <param name="_ExtentX" value ="2646">
    <param name="_ExtentY" value ="1323">
    <param name="_StockProps" value ="0">
</object>
</center>
<center>

<form name="form">
<!-- Destination URL of connection -->
URL <input name="URL" type="TEXT" value="rtsp://192.168.0.250/" ><BR>
<!-- Video Format -->
VIDEO
<input name="Video" type="RADIO" onclick=RADIO_OnClick() value="MPEG4">MPEG4
<input name="Video" type="RADIO" onclick=RADIO_OnClick() value="MJPEG" checked>MJPEG<BR>
<!-- Connection -->
<input name = "Connect" type = "button" value = "Connect">
<!-- Disconnection -->
<input name = "Disconnect" type = "button" value = "Disconnect">
<!-- Mute -->
<INPUT NAME = "Mute" TYPE = "checkbox" VALUE = "Mute">Mute
<!-- Send audio data -->
<INPUT NAME = "Send" TYPE = "checkbox" VALUE = "Send">Send

</center>
```

```
</form>

<script language = "VBScript">
<!--
Dim TheForm

Sub RADIO_OnClick   'Setting for Video Format
    Set TheForm = Document.form
    If TheForm.elements(1).checked = True Then
        NSPlayer1.VideoFormat = 1    'Set MPEG
    End if
    If TheForm.elements(2).checked = True Then
        NSPlayer1.VideoFormat = 0    'Set MJPEG
    End If
End Sub

Sub Connect_OnClick 'Connection
    Set TheForm = Document.form
    call NSPlayer1.Start(TheForm.URL.Value) 'Connection
End Sub

Sub Disconnect_OnClick  'Disconnection
    call NSPlayer1.Stop()   'Disconnection
End Sub

Sub Mute_OnClick    'Mute
    Set TheForm = Document.form
    If TheForm.Mute.checked = True Then
        call NSPlayer1.Mute(TRUE)   'Mute ON
    End if
    If TheForm.Mute.checked = False Then
        call NSPlayer1.Mute(FALSE)  'Mute OFF
    End if
End Sub

Sub Send_OnClick    'Send Audio data
    Set TheForm = Document.form
    If TheForm.Send.checked = True Then
        call NSPlayer1.Mic(TRUE)    'Start to send Audio data
    End if
    If TheForm.Send.checked = False Then
        call NSPlayer1.Mic(FALSE)   'Finish sending Audio data
    End if
End Sub
-->
</script>

</body>
</html>
```