Developer World
developer.sony.com

February 2014

API references for Camera Remote API beta

Camera Remote API API Reference

First edition (December 2013)

Preface

Developer World

For the latest Sony technical news, tutorials and development tools go to developer.sony.com

About this document

The “Camera Remote API beta” will be referred to as the “Camera Remote API” in this document.
The purpose of this document is to list the API specifications for the Camera Remote API provided by Sony.

Document conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

HTTP	HyperText Transfer Protocol (RFC 2616)
JSON	JavaScript Object Notation (RFC 4627)
JSON-RPC	Remote Procedure Call encoded in JSON
SSDP	Simple Server Discovery Protocol

Document history

Change history		
2013-09-01	Version 1.0	First version
2013-09-25	Version 1.10	Added 6 compatible cameras to section 2.3. Modified "API name parameters" in section 3.11 to support more compatible cameras. Corrected some typographical errors.
2013-11-05	Version 1.20	Added 3 compatible cameras to section 2.3. Added 2 audio recording APIs. Added "audio" shoot mode parameter.
2013-12-01	Version 1.21	Updated to new template
2014-01-06	Version 1.30	Added 1 compatible camera. Added zoom support to the PlayMemories Camera Apps compatible cameras.
2014-02-18	Version 1.40	Added 4 compatible cameras.

Contents

Introduction

The purpose of this document is to describe the API specifications for the Camera Remote API. If you want to get information about how to access camera functions and the procedure to establish connection to use the APIs, please see the Camera Remote API Development Guide available in the Camera Remote API SDK.

API versioning, supported and available APIs

Camera Remote API versioning

Camera Remote API includes versioning on two levels. The Camera Remote API itself has one version, and then each API has their own version. The client app may check those versions and change its behavior accordingly.

Camera Remote API version

Camera Remote API has its version defined by its specifying functions. The version will be changed if an API was added or deleted. The version also will be changed if a supporting function in any APIs was changed. The Camera Remote API version can be obtained by the "getApplicationInfo" API. For details, please see the “getApplicationInfo" API specification.

This document only cover Camera Remote API version 2.0.0 or greater. The client app should check whether the API version is “2.0.0" or greater to confirm the server function compatibility with this document.

Version for each API

Each API available in the Camera Remote API has its version also. This version is represented as two numbers separated by "."(dot), X.Y. Basically, every API will be defined from version "1.0". This number will be incremented when the definition of the API is changed.

The client app can get what versions a camera supports by using the "getVersions" API. The versions received from the “getVersions" API depend on what APIs are supported on an API service of that camera and depend on the camera’s API capability. **The client app MUST set the "version" parameter in the request to specify the version of the API.** If you want to know what versions each API supports, please see each API specification.

Supported APIs and available APIs

The camera functions can vary between models. Therefore the APIs that each camera supports may vary by models. The camera provides its supported API list via the "getMethodTypes" API.

In addition, the camera status can be changed by user operations and calling APIs. Available APIs in the camera will be changed by camera status. The camera also provides a list of available APIs via the "getAvailableApiList" API or the "availableApiList" object of the "getEvent" API callback. For more information, please see API specification.

Supported API groups for each compatible cameras

	HDR-AS15 *2	HDR-AS30 HDR-AS100	HDR-MV1	A7R *3 A7 *3 NEX-6 *3 NEX-5R *3 NEX-5T *3 A5000 *3 A6000 *3 DSC-HX60 *3 DSC-HX400 *3	DSC-QX100 DSC-QX10
Shoot mode	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Still capture	No	**Yes** *4	No	**Yes**	**Yes** *4
Movie recording	**Yes**	**Yes**	**Yes**	No	**Yes**
Audio Recording	No	No	**Yes**	No	No
Liveview	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Zoom	No	No	No	**Yes**	**Yes**
Self-timer	No	No	No	**Yes**	**Yes**
Postview image size	No	No	No	**Yes**	**Yes**
Event notification	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Camera setup *1	No	No	No	**Yes**	No
Server information	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

API groups - Compatible cameras

*1: Some camera models need "Camera setup" API call before accessing camera remote shooting functions.

*2: The latest firmware update is needed.

*3: These cameras are compatible with the PlayMemories Camera Apps "Smart Remote Control" application.
The latest version of the application should be installed and started to use the APIs.

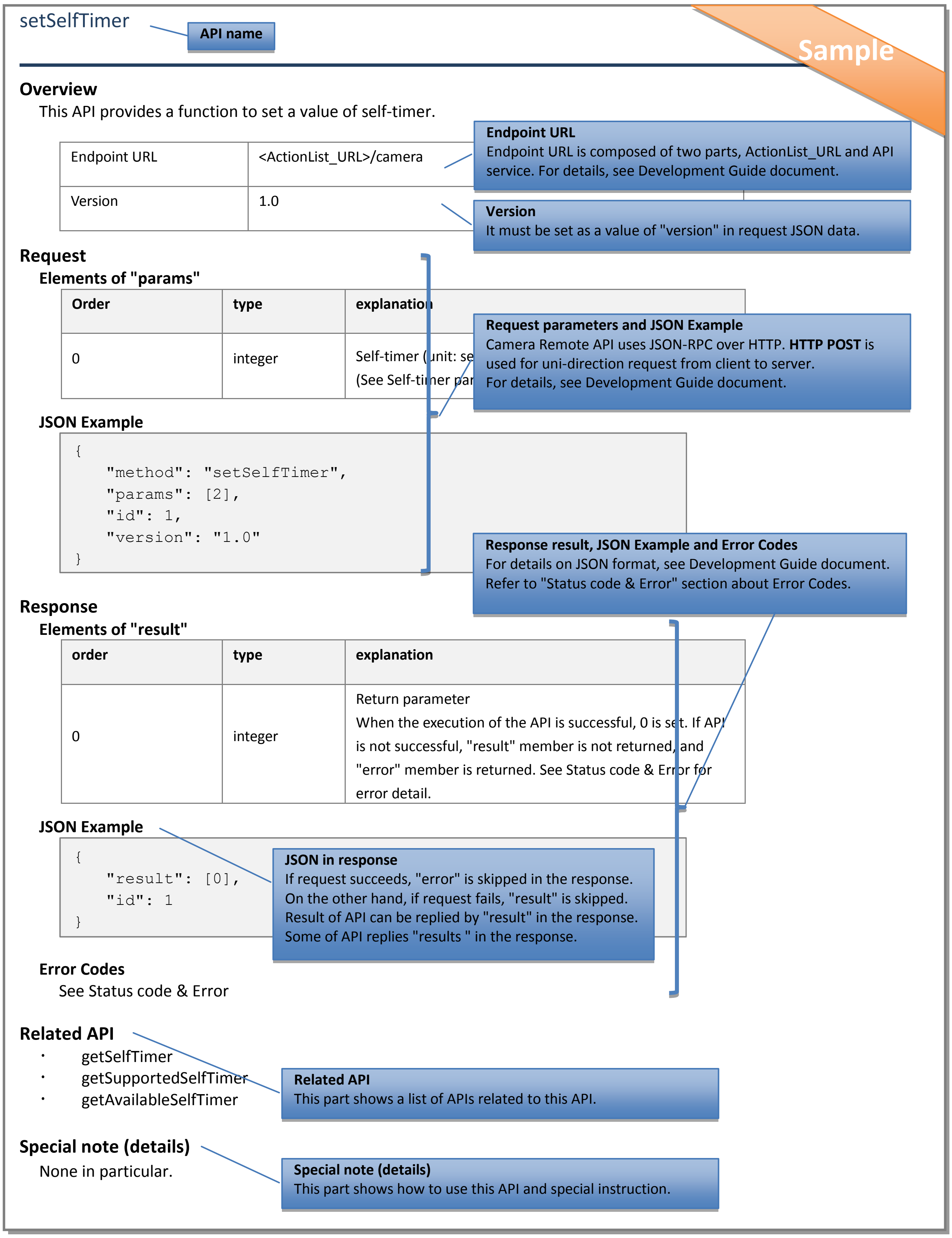
*4: These cameras support only "actTakePicture".

API list

API groups	APIs
Shoot mode	setShootMode
	getShootMode
	getSupportedShootMode
	getAvailableShootMode
Still capture	actTakePicture
	awaitTakePicture
Movie recording	startMovieRec
	stopMovieRec
Audio recording	startAudioRec
	stopAudioRec
Liveview	startLiveview
	stopLiveview
Zoom	actZoom
Self-timer	setSelfTimer
	getSelfTimer
	getSupportedSelfTimer
	getAvailableSelfTimer
Postview image size	setPostviewImageSize
	getPostviewImageSize
	getSupportedPostviewImageSize
	getAvailablePostviewImageSize
Event notification	getEvent
Camera setup	startRecMode
	stopRecMode
Server information	getAvailableApiList
	getApplicationInfo
	getVersions
	getMethodTypes

API Reference

This chapter provides the detailed API specification of Camera Remote API using the below format.



setSelfTimer

API name

Overview

This API provides a function to set a value of self-timer.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Endpoint URL
Endpoint URL is composed of two parts, ActionList_URL and API service. For details, see Development Guide document.

Version
It must be set as a value of "version" in request JSON data.

Request

Elements of "params"

Order	type	explanation
0	integer	Self-timer (unit: se (See Self-timer par

Request parameters and JSON Example
Camera Remote API uses JSON-RPC over HTTP. **HTTP POST** is used for uni-direction request from client to server.
For details, see Development Guide document.

JSON Example

```
{
    "method": "setSelfTimer",
    "params": [2],
    "id": 1,
    "version": "1.0"
}
```

Response result, JSON Example and Error Codes
For details on JSON format, see Development Guide document.
Refer to "Status code & Error" section about Error Codes.

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

JSON in response
If request succeeds, "error" is skipped in the response.
On the other hand, if request fails, "result" is skipped.
Result of API can be replied by "result" in the response.
Some of API replies "results " in the response.

Error Codes

See Status code & Error

Related API

- getSelfTimer
- getSupportedSelfTimer
- getAvailableSelfTimer

Related API
This part shows a list of APIs related to this API.

Special note (details)

None in particular.

Special note (details)
This part shows how to use this API and special instruction.

Shoot mode

setShootMode

Overview

This API provides a function to set a value of shooting mode.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

order	type	explanation
0	string	Shoot mode (See Shoot mode parameters of Parameter description)

JSON Example

```
{
    "method": "setShootMode",
    "params": ["movie"],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- getShootMode
- getSupportedShootMode
- getAvailableShootMode

Special note (details)

The camera has the concept of shoot mode. Some of APIs are only available on specific shoot mode. For details, see "Supported APIs and available APIs".

Even if the response is successful, the setting may not be finished on the server. Therefore, the client can check "shootMode" object in "getEvent" callback to recognize the timing of a change in the parameter of the server.

Some camera models need "startRecMode" API call before accessing camera settings. See "startRecMode" for details.

getShootMode

Overview

This API provides a function to get current camera shooting mode.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getShootMode",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string	Current shoot mode (See Shoot mode parameters of Parameter description)

JSON Example

```
{
    "result": ["still"],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setShootMode
- getSupportedShootMode
- getAvailableShootMode

Special note (details)

None in particular.

getSupportedShootMode

Overview

This API provides a function to get the supported shoot modes.
The client should use "getAvailableShootMode" to get the available parameters at the moment.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getSupportedShootMode",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string-array	A list of supported shoot modes (See Shoot mode parameters of Parameter description)

JSON Example

```
{
    "result": [
        ["still","movie"]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setShootMode
- getShootMode
- getAvailableShootMode

Special note (details)

This method returns the same value for the response parameter only when this method is successful and the function is available. The result of the response depends on camera models.

getAvailableShootMode

Overview

This API provides a function to get current shoot mode and the available shoot modes at the moment. The available parameters can be changed by user operations and calling APIs.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getAvailableShootMode",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

Order	type	explanation
0	string	Current shoot mode (See Shoot mode parameters of Parameter description)
1	string-array	A list of available shoot modes (See Shoot mode parameters of Parameter description)

JSON Example

```
{
    "result": [
        "still",
        ["still","movie"]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setShootMode
- getShootMode
- getSupportedShootMode

Special note (details)

This API returns current set value and available values at the moment. The client can get the same response of this API via "shootMode" object in "getEvent" callback.

Still capture

actTakePicture

Overview

This API provides a function to take picture.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "actTakePicture",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string-array	Array of URLs of postview. The postview is captured image data by camera. The postview image can be used for storing it as the taken picture, and showing it to the client display.

JSON Example

```
{
    "result": [
        ["http://ip:port/postview/postview.jpg"]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- awaitTakePicture
- getEvent

Special note (details)

This API instructs the server side to shoot still image. When this API is called and the server starts shooting still image, the camera status will change as follows. The camera status can be obtained by "getEvent".

Camera status: "IDLE" -> "StillCapturing" -> "StillSaving" -> "IDLE"

Note that this sequence is the example of typical case.

The client should check the "getEvent" parameter ("cameraStatus") and check if it is "IDLE" before calling this API.

In case of long exposure, the server will return "40403" error ("Still Capturing Not Finished") within several tens of seconds. If status code "40403" is received, capturing is not completed. Use the "awaitTakePicture" API to receive status on capture. If status code "40403" is received for "awaitTakePicture" again, the client can call "awaitTakePicture" until the capture is done.

This API is only available when the shoot mode is "still".

Some camera models need "startRecMode" API call before capturing still image. See "startRecMode" for details.

awaitTakePicture

Overview

This API provides a function to wait while the camera is taking the picture.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "awaitTakePicture",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string-array	Array of URLs of postview. The postview is captured image data by camera. The postview image can be used for storing it as the taken picture, and showing it to the client display.

JSON Example

```
{
    "result": [
        ["http://ip:port/postview/postview.jpg"]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- actTakePicture
- getEvent

Special note (details)

Please see "actTakePicture".

Movie recording

startMovieRec

Overview

This API provides a function to start recording movie.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "startMovieRec",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- stopMovieRec
- getEvent

Special note (details)

This API instructs the server side to start recording movie. When this API is called and the server starts recording movie, the camera status will change as follows. The camera status can be obtained by "getEvent".

[Client calls "startMovieRec"]

Camera status: "IDLE" -> "MovieWaitRecStart" -> "MovieRecording".

After the recording has started, the client may stop the recording. To stop the recording, "stopMovieRec" must be called, and the camera status will change as follows.

[Client calls "stopMovieRec"]

Camera status: "MovieRecording" -> "MovieWaitRecStop" -> "MovieSaving" -> "IDLE".

Note that this sequence is the example of typical case. For example, some servers may skip "MovieWaitRecStop".

The client should check the "getEvent" parameter ("cameraStatus") and check if it is "IDLE" before calling "startMovieRec".

This API is only available when the shoot mode is "movie".
Some camera models need "startRecMode" API call before recording movie. See "startRecMode" for details.

stopMovieRec

Overview

This API provides a function to stop recording movie.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "stopMovieRec",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string	URL of thumbnail (If the server does not support, empty string will return.

JSON Example

```
{
    "result": [
        ""
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- startMovieRec

Special note (details)

This API is only available when the shoot mode is "movie".

Even if this API is successful, the server may not be ready to start the next shot. The next shot is prohibited until the client could make sure, that the server is ready to start the next shot, through the "getEvent" callback parameter "cameraStatus". See "startMovieRec" API specification for the detail.

The server may return null or empty string for the parameter "URL of thumbnail" depending on camera models. So the client must ignore the value if it is returned with null or empty string.

Audio recording

startAudioRec

Overview

This API provides a function to start audio recording.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "startAudioRec",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- stopAudioRec
- getEvent

Special note (details)

This API instructs the server side to start audio recording. When this API is called and the server starts audio recording, the camera status will change as follows. The camera status can be obtained by "getEvent".

[Client calls “startAudioRec"]

Camera status: "IDLE" -> "AudioWaitRecStart" -> "AudioRecording".

After the recording has started, the client may stop the recording. To stop the recording, "stopAudioRec” must be called, and the camera status will change as follows.

[Client calls "stopAudioRec”]

Camera status: "AudioRecording" -> "AudioWaitRecStop" -> "AudioSaving" -> "IDLE".

Note that this sequence is the example of typical case.

The client should check the "getEvent" parameter ("cameraStatus") and check if it is "IDLE" before calling “startAudioRec".

This API is only available when the shoot mode is "audio".

Note that the server may disable the liveview function when the shoot mode is "audio". The client should check liveview availability by "liveviewStatus" of "getEvent". The APIs availability will also be changed. The client should check the APIs availability by available API list. When the client switches the shoot mode from "audio" to others, the client can restart the liveview by calling "startLiveview".

stopAudioRec

Overview

This API provides a function to stop audio recording.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "stopAudioRec",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, “result” member is not returned, and “error” member is return. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0]
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- startAudioRec

Special note (details)

This API is only available when the shoot mode is "audio".

Even if this API is successful, the server may not be ready to start the next recording. The next recording is prohibited until the client could make sure, that the server is ready to start the next recording, through the "getEvent" callback parameter "cameraStatus".

Liveview

startLiveview

Overview

This API provides a function to start liveview.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "startLiveview",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string	URL of liveview

JSON Example

```
{
    "result": [
        "http://ip:port/liveview/liveviewstream"
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- stopLiveview

Special note (details)

This API commands the server to start the liveview function. When this API is successful, the client can obtain the URL for downloading the liveview data. The method to obtain liveview is HTTP GET. The liveview comes in one data stream. Refer to Liveview Data Format.

The server may stop liveview by itself. The client should check the status of the liveview, which can be obtained by "getEvent" callback parameter "liveviewStatus". When this API is called and the server cannot start liveview, this API returns the callback with error. The client should wait for an appropriate interval, and then recall this API and restart. Some camera models need "startRecMode" API call before starting the liveview. See "startRecMode" for details.

Please see Sample Sequence for more information about the procedure of calling APIs.

stopLiveview

Overview

This API provides a function to stop liveview.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "stopLiveview",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- startLiveview

Special note (details)

This API instructs the server side to stop the liveview function.

After calling this API, the client should not access the URL obtained in "startLiveview" before. If the client would like to start liveview again, it should obtain the liveview URL again by calling "startLiveview".

Zoom

actZoom

Overview

This API provides a function to zoom.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

order	type	explanation
0	string	Direction (See Zoom parameters of Parameter description)
1	string	Movement (See Zoom parameters of Parameter description)

JSON Example

```
{
    "method": "actZoom",
    "params": ["in","start"],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

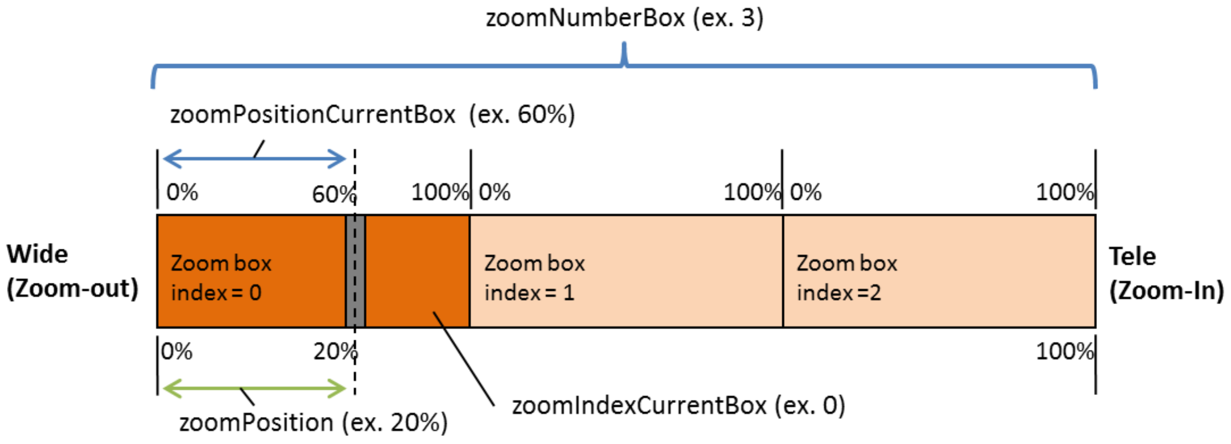
- getEvent

Special note (details)

When the client set "stop" as movement parameter, the direction should be the same as the last "start". For example, if the client set "in"-"start" at the beginning, "in"-"stop" are necessary. If the client set "in"-"start", and set "out"-"stop" later, then the callback will include error.

When the client set "start" as movement parameter, zoom operation will be continued until "stop" or reaching the termination. When "1shot" is set, zoom operation will be stopped after a certain position.

The client can check the zoom information using "getEvent". The zoom information consists of four parameters, "zoomPosition", "zoomNumberBox", "zoomIndexCurrentBox", and "zoomPositionCurrentBox" as follows. "Zoom box" represents the type of zoom such as optical and digital.



type	name	explanation
integer	zoomPosition	Zoom position to the whole (0 - 100, unit: percentage)
integer	zoomNumberBox	Number of zoom box
integer	zoomIndexCurrentBox	Index of current zoom box (starts from 0)
integer	zoomPositionCurrentBox	Zoom position in the current zoom box (0 - 100, unit: percentage)

Self-timer

setSelfTimer

Overview

This API provides a function to set a value of self-timer.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

order	type	explanation
0	integer	Self-timer (unit: second) (See Self-timer parameters of Parameter description)

JSON Example

```
{
    "method": "setSelfTimer",
    "params": [2],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- getSelfTimer
- getSupportedSelfTimer
- getAvailableSelfTimer

Special note (details)

Even if the response is successful, the setting may not be finished on the server. Therefore, the client can check "selfTimer" object in "getEvent" callback to recognize the timing of a change in the parameter of the server.

Some camera models need "startRecMode" API call before accessing camera settings. See "startRecMode" for details.

getSelfTimer

Overview

This API provides a function to get current self-timer setting.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getSelfTimer",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Current self-timer setting (unit: second) (See Self-timer parameters of Parameter description)

JSON Example

```
{
    "result": [2],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setSelfTimer
- getSupportedSelfTimer
- getAvailableSelfTimer

Special note (details)

None in particular.

getSupportedSelfTimer

Overview

This API provides a function to get the supported self-timer settings.
The client should use "getAvailableSelfTimer" to get the available parameters at the moment.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getSupportedSelfTimer",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer-array	A list of supported self-timer settings (unit: second) (See Self-timer parameters of Parameter description)

JSON Example

```
{
    "result": [
        [0,2,10]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setSelfTimer
- getSelfTimer
- getAvailableSelfTimer

Special note (details)

This method returns the same value for the response parameter only when this method is successful and the function is available. The result of the response depends on camera models.

getAvailableSelfTimer

Overview

This API provides a function to get current self-timer setting and the available self-timer settings at the moment. The available parameters can be changed by user operations and calling APIs.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getAvailableSelfTimer",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Current self-timer setting (unit: second) (See Self-timer parameters of Parameter description)
1	integer-array	A list of available self-timer settings (unit: second) (See Self-timer parameters of Parameter description)

JSON Example

```
{
    "result": [
        0,
        [0,2,10]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setSelfTimer
- getSelfTimer
- getSupportedSelfTimer

Special note (details)

This API returns current set value and available values at the moment. The client can get the same response of this API via "selfTimer" object in "getEvent" callback.

Postview image size

setPostviewImageSize

Overview

This API provides a function to set a value of postview image size. The postview image can be used for storing it as the taken picture, and showing it to the client display.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

order	type	explanation
0	string	Postview image size (See Postview image size parameters of Parameter description)

JSON Example

```
{
    "method": "setPostviewImageSize",
    "params": ["Original"],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- getPostviewImageSize
- getSupportedPostviewImageSize
- getAvailablePostviewImageSize

Special note (details)

The postview is the still image data that can be received as the response of "actTakePicture" and "awaitTakePicture".
Even if the response is successful, the setting may not be finished on the server. Therefore, the client can check "postviewImageSize" object in "getEvent" callback to recognize the timing of a change in the parameter of the server.

Some camera models need "startRecMode" API call before accessing camera settings. See "startRecMode" for details.

getPostviewImageSize

Overview

This API provides a function to get current postview image size.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getPostviewImageSize",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string	Current postview image size (See Postview image size parameters of Parameter description)

JSON Example

```
{
    "result": [
        "Original"
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setPostviewImageSize
- getSupportedPostviewImageSize
- getAvailablePostviewImageSize

Special note (details)

None in particular.

getSupportedPostviewImageSize

Overview

This API provides a function to get the supported postview image sizes.
The client should use "getAvailablePostviewImageSize" to get the available parameters at the moment.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getSupportedPostviewImageSize",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string-array	A list of supported postview image sizes (See Postview image size parameters of Parameter description)

JSON Example

```
{
    "result": [
        ["Original","2M"]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setPostviewImageSize
- getPostviewImageSize
- getAvailablePostviewImageSize

Special note (details)

This method returns the same value for the response parameter only when this method is successful and the function is available. The result of the response depends on camera models.

getAvailablePostviewImageSize

Overview

This API provides a function to get current postview image size and the available postview image sizes at the moment. The available parameters can be changed by user operations and calling APIs.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getAvailablePostviewImageSize",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string	Current postview image size (See Postview image size parameters of Parameter description)
1	string-array	A list of available postview image sizes (See Postview image size parameters of Parameter description)

JSON Example

```
{
    "result": [
        "Original",
        ["Original","2M"]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- setPostviewImageSize
- getPostviewImageSize
- getSupportedPostviewImageSize

Special note (details)

This API returns current set value and available values at the moment. The client can get the same response of this API via "postviewImageSize" object in "getEvent" callback.

Event notification

getEvent

Overview

This API provides a function to get event from the server.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

order	type	explanation
0	boolean	Long polling flag true: Callback when timeout or change point detection. false: Callback immediately.

JSON Example

```
{
    "method": "getEvent",
    "params": [true],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	Key	type	explanation
0		object	Available API list
	"type"	string	Name of object type ("availableApiList")
	"names"	string-array	A list of available API names (See API name parameters and getAvailableApiList)
1		object	Camera status
	"type"	string	Name of object type ("cameraStatus")
	"cameraStatus"	string	Camera status (See Event parameters)
2		object	Zoom information
	"type"	string	Name of object type ("zoomInformation")
	"zoomPosition"	integer	Zoom position to the whole (See actZoom) (When -1 is set, this parameter is invalid.)
	"zoomNumberBox"	integer	Number of zoom box (See actZoom) (When -1 is set, this parameter is invalid.)
	"zoomIndexCurrentBox"	integer	Index of current zoom box (See actZoom) (When -1 is set, this parameter is invalid.)
	"zoomPositionCurrentBox"	integer	Zoom position in the current zoom box (See actZoom) (When -1 is set, this parameter is invalid.)

3		object	Liveview status
	"type"	string	Name of object type ("liveviewStatus")
	"liveviewStatus"	boolean	Liveview status true: Ready to transfer Liveview images. false: Not ready to transfer Liveview images.
4 to 18		object/array	Reserved
19		object	Postview image size
	"type"	string	Name of object type ("postviewImageSize")
	"currentPostviewImageSize"	string	Current postview image size (See Postview image size parameters)
	"postviewImageSizeCandidates"	string-array	A list of available postview image sizes (See Postview image size parameters)
20		object	Self-timer
	"type"	string	Name of object type ("selfTimer")
	"currentSelfTimer"	integer	Current self-timer setting (unit: second) (See Self-timer parameters) (When -1 is set, this parameter is invalid.)
	"selfTimerCandidates"	integer-array	A list of available self-timer settings (unit: second) (See Self-timer parameters)
21		object	Shoot mode
	"type"	string	Name of object type ("shootMode")
	"currentShootMode"	string	Current shoot mode (See Shoot mode parameters)
	"shootModeCandidates"	string-array	A list of available shoot modes (See Shoot mode parameters)

JSON Example

```
{
    "result": [
        {
            "type":"availableApiList",
            "names":["startLiveview","stopLiveview","setSelfTimer",...]
        },
        {
            "type":"cameraStatus",
            "cameraStatus":"IDLE"
        },
        {
            "type":"zoomInformation",
            "zoomPosition":0,
            "zoomNumberBox":1,
            "zoomIndexCurrentBox":0,
            "zoomPositionCurrentBox":0
        },
        {
            "type":"liveviewStatus",
            "liveviewStatus":true
        },
        ...,
        {
            "type":"postviewImageSize",
            "currentPostviewImageSize":"2M",
            "postviewImageSizeCandidates":["Original","2M"]
        },
        {
            "type":"selfTimer",
            "currentSelfTimer":0,
            "selfTimerCandidates":[0,2,10]
        },
        {
            "type":"shootMode",
            "currentShootMode":"still",
            "shootModeCandidates":["still","movie"]
        },
        ...
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

None.

Special note (details)

The purpose of this API is sending the event from the server actively. When the client calls some APIs and/or the user operates the camera directly, the client can get the parameter updates from the camera by using this API.

If getEvent is executed, the server times out or does not return a response until the parameter is updated. (If the input parameter "polling" is false, the server replies immediately.) If the client gets the response, the client should execute getEvent immediately. When the execution of this API is unsuccessful or timeout, the server will notify error in response.

When each object in the response is null or empty array as JSON, it means that no update was happened in the server for the object type or the server does not support the object type. In case of empty string or empty array in supported object, it means that the parameter is invalid. In regard to some objects related to getAvailableXXX API, the client can check if the object is valid via "getAvailableApiList" or "availableApiList" object.

The callback parameter value could be the same as the previous one, so the client should evaluate the value of callback. The client can detect each object in the callback using "Name of object type" which is in each object. The client must ignore objects which are not described in this document.

In case this API is called with "polling=true", the server will not return a response until the server is ready, and any value has changed in the server. So, by calling with polling=true, the client can recognize the timing of a change in the parameter of the server. For each parameter, if the current value or available candidates doesn't update, the parameter object including "type" will be null object or empty array. Otherwise, the object will be notified. This API with "polling=true" is a notifying type method and acts with other requiring type methods in parallel.

In case this API is called with "polling=false", the server will return a response immediately. So, by calling with polling=false, the client can recognize the server state snapshot at the time. The client can call this API with "polling=false" at the beginning. (The information obtained in the response may be useful for the client to build UI layouts and so on.)

For more information about how to use this API, see Sample Sequence.

Camera setup

startRecMode

Overview

This API provides a function to set up camera for remote shooting. Some camera models need this API call before starting liveview, capturing still image, recording movie, or accessing all other camera remote shooting functions.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "startRecMode",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- stopRecMode

Special note (details)

Some camera models need this API call before starting liveview, capturing still image, recording movie, or accessing all other camera remote shooting functions. The client must check if the server needs this API call. The check can be done by checking the availability of this API in "getAvailableApiList" or "getMethodTypes" callback. The client should call this API just once before accessing camera remote shooting functions if the server needs.

stopRecMode

Overview

This API provides a function to stop remote shooting functions.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "stopRecMode",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	integer	Return parameter When the execution of the API is successful, 0 is set. If API is not successful, "result" member is not returned, and "error" member is returned. See Status code & Error for error detail.

JSON Example

```
{
    "result": [0],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- startRecMode

Special note (details)

None in particular.

Server information

getAvailableApiList

Overview

This API provides a function to get the available API names that the server supports at the moment.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getAvailableApiList",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string-array	A list of available API names (See API name parameters of Parameter description)

JSON Example

```
{
    "result": [
        ["startLiveview","stopLiveview","setSelfTimer",...]
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- getMethodTypes

Special note (details)

The client gets the list of API names that can be executed at the moment from callback parameter. The callback parameter "A list of available API names" varies depending on the camera status at the moment. For example, when the server is in "movie" mode, "setSelfTimer" will not be in the callback parameter.

The client should ignore API names which are not described in this document.

getApplicationInfo

Overview

This API provides a function to get name and "Camera Remote API" version of the server.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getApplicationInfo",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string	Application name of the server
1	string	"Camera Remote API" version of the server

JSON Example

```
{
    "result": [
        "Smart Remote Control",
        "2.0.0"
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- getVersions

Special note (details)

The client should call this API during initialization sequence. The client can get the following "Camera Remote API" version of server. The version definition: "M.m.x".

[M]: Major number (Any number)
[m]: Minor number (Any number)
[x]: Server release number (Any number)

The client should check whether the API version is "2.x.x" ("2.0.0" or greater) to confirm the server function compatibility with this document.

getVersions

Overview

This API provides supported versions on the "API service". The client can get the list of API names for specific version using "getMethodTypes" API. The client can get list of versions, which the server supports, using this API.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

Empty.

JSON Example

```
{
    "method": "getVersions",
    "params": [],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "result"

order	type	explanation
0	string-array	Supported versions on the API service.

JSON Example

```
{
    "result": [
       ["1.0"]
   ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- getMethodTypes

Special note (details)

The version in the response is defined for each API in the "API service". For more information about "API service", see Development Guide. When the client calls the APIs to use camera functionality, API service is basically "camera".

getMethodTypes

Overview

This API provides a function to get the supported APIs for the version. The client can get the list of API names for specific version using this API. The client can get list of versions, which the server supports, using "getVersions" API.

Endpoint URL	<ActionList_URL>/camera
Version	1.0

Request

Elements of "params"

order	type	explanation
0	string	Version for each API. In case this parameter is empty string, response includes methods of all versions supported in camera.

JSON Example

```
{
    "method": "getMethodTypes",
    "params": ["1.0"],
    "id": 1,
    "version": "1.0"
}
```

Response

Elements of "results"

order		type	explanation
0 to N		array	Array of elements
	0	string	API name
	1	string-array	Array of parameter types
	2	string-array	Array of response types
	3	string	Version for each API

JSON Example

```
{
    "results": [
        ["getAvailableApiList",[],["string*"],"1.0"],
        ["setShootMode",["string"],["int"],"1.0"],
        ...
    ],
    "id": 1
}
```

Error Codes

See Status code & Error

Related API

- getVersions

Special note (details)

The definition of "parameter types" is below.

value	explanation
"bool"	True or false value is stored.
"bool*"	Multiple true or false values are stored in an array.
"int"	Integer number.
"int*"	Multiple integer numbers are stored in an array.
"double"	Double number.
"double*"	Multiple double numbers are stored in an array.
"string"	String data is stored.
"string*"	Multiple string data are stored in an array.
"{"object name":"object type"}"	An Object is stored. Double quotes in object are escaped to store object structure into a string.
"{"object name":"object type"}*"	Multiple objects (of the same type) are stored in an array. Double quotes in object are escaped to store object structure into a string.

Most of APIs will be replied by "result" in those responses. Note that result of this API will be replied by "results" in the response.

The camera status can be changed by user operations and calling APIs. Available APIs in the camera will be changed by camera status. Therefore, the client should check available APIs at the moment via "getAvailableApiList" API or "availableApiList" object of "getEvent" API callback.

The client should ignore API names which are not described in this document.

Parameter description

API name parameters

value	explanation
"setShootMode"	See "setShootMode"
"getShootMode"	See "getShootMode"
"getSupportedShootMode"	See "getSupportedShootMode"
"getAvailableShootMode"	See "getAvailableShootMode"
"startLiveview"	See "startLiveview"
"stopLiveview"	See "stopLiveview"
"actZoom"	See "actZoom"
"setSelfTimer"	See "setSelfTimer"
"getSelfTimer"	See "getSelfTimer"
"getSupportedSelfTimer"	See "getSupportedSelfTimer"
"getAvailableSelfTimer"	See "getAvailableSelfTimer"
"setPostviewImageSize"	See "setPostviewImageSize"
"getPostviewImageSize"	See "getPostviewImageSize"
"getSupportedPostviewImageSize"	See "getSupportedPostviewImageSize"
"getAvailablePostviewImageSize"	See "getAvailablePostviewImageSize"
"getEvent"	See "getEvent"
"startRecMode"	See "startRecMode"
"stopRecMode"	See "stopRecMode"
"getAvailableApiList"	See "getAvailableApiList"
"getApplicationInfo"	See "getApplicationInfo"
"getVersions"	See "getVersions"
"getMethodTypes"	See "getMethodTypes"

Note that the client can check the availability of "actTakePicture", "awaitTakePicture", "startMovieRec", "stopMovieRec", “startAudioRec” or “stopAudioRec” by checking current shooting mode. Current shooting mode can be obtained by "getShootMode", "getAvailableShootMode" or "getEvent".

Shoot mode parameters

value	explanation
"still"	Still image shoot mode
"movie"	Movie shoot mode
“audio”	Audio shoot mode

Self-timer parameters

value	explanation
0	Off
2	2 seconds
10	10 seconds

Zoom parameters

Zoom direction parameter

Value	explanation
"in"	Zoom-In
"out"	Zoom-Out

Zoom movement parameter

Value	explanation
"start"	Long push
"stop"	Stop
"1shot"	Short push

Postview image size parameters

value	explanation
"Original"	Original size
"2M"	2M-pixel size

Event parameters

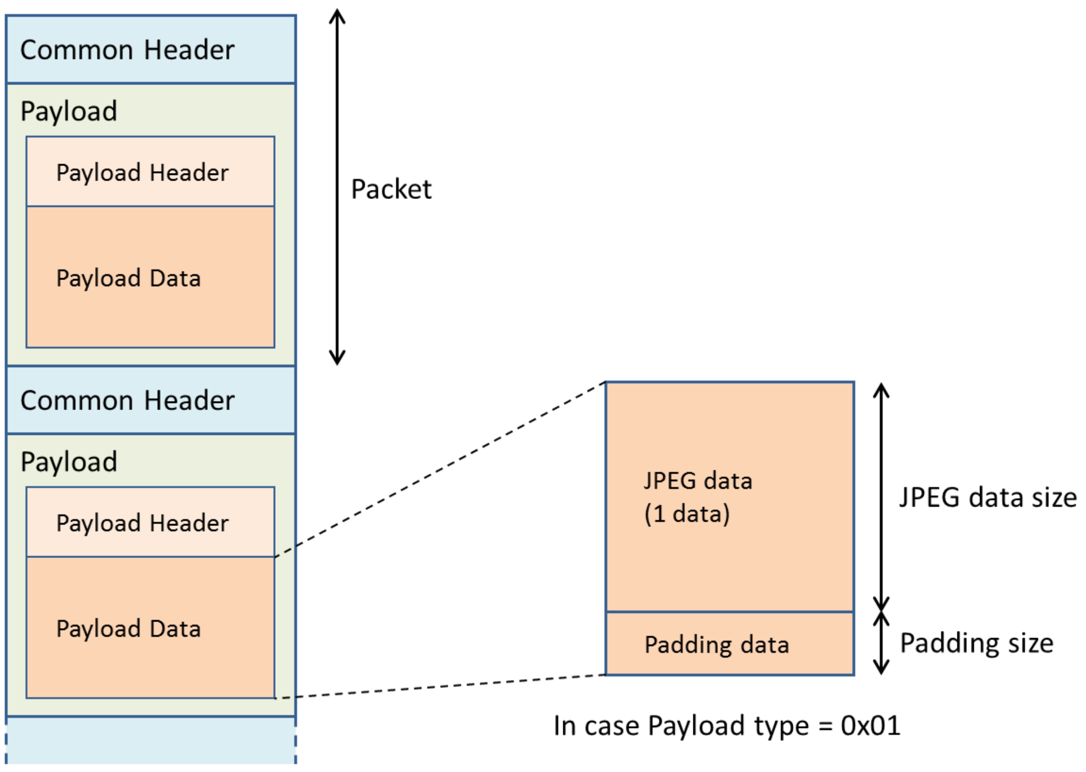
Camera status parameters

value	explanation
"Error"	Error at the server (ex. high temperature, no memory card)
"NotReady"	The server cannot start recording (ex. during initialization, mode transitioning)
"IDLE"	Ready to record
"StillCapturing"	Capturing still images
"StillSaving"	Saving still images
"MovieWaitRecStart"	Preparing to start recording movie
"MovieRecording"	Recording movie
"MovieWaitRecStop"	Stopping the movie recording
"MovieSaving"	Saving movie
"AudioWaitRecStart"	Preparing to start recording audio
"AudioRecording"	Recording audio
"AudioWaitRecStop"	Stopping the audio recording
"AudioSaving"	Saving audio

Liveview data format

Format of the liveview data JPEG container

- ✓ The liveview data is downloaded as one data stream, by HTTP GET.
  - The smallest unit is called "Packet" as the diagram below indicates. During the download, this "Packet" will be repeated.
  - The client should keep on downloading the data of "Packet", and extract JPEG image from one "Packet", and decode it, and show it to the display.
  - The client may not display all the JPEG because of the decoding time. In that case, the client should skip some JPEG images.
- ✓ The endian of the data is network byte order.
- ✓ Refer to the following sections for more information about Common Header and Payload.



Common Header

Common Header is constructed of the following 8 Bytes.

- ✓ Start byte : 1 [B]
  - 0xFF, fixed
- ✓ Payload type : 1 [B]
  - indicates type of the Payload
  - 0x01 = For liveview images
- ✓ Sequence number : 2 [B]
  - Frame No, 2 bytes integer and increments every frame
  - This frame no will be repeated.
- ✓ Time stamp : 4 [B]
  - 4 bytes integer, the unit will be indicated by Payload type
  - In case Payload type = 0x01, the unit of the Time stamp of the Common Header is milliseconds. The start time may not start from zero and depends on the server.

Payload Header

In case Payload type = 0x01, the header format will be as following 128 Bytes.

- ✓ Start code : 4[B]
  - fixed (0x24, 0x35, 0x68, 0x79)
  - This can be used for detection of the payload header.
- ✓ JPEG data size : 3[B]
  - Size of JPEG in Payload data, Bytes.
- ✓ Padding size : 1[B]
  - Padding size of the Payload data after the JPEG data, Bytes.
- ✓ Reserved : 4[B]
- ✓ Flag : 1[B]
  - 0x00, fixed.
- ✓ Reserved : 115[B]

Payload Data

In case Payload type = 0x01

- ✓ JPEG data(1 data)
  - This size is indicated as "JPEG data size" in Payload Header.
- ✓ Padding data
  - This size is indicated as "Padding size" in Payload Header (No padding if 0 bytes was indicated).

Status code & Error

Major status codes are below. The "error" member is defined as [error_code, error_message].
The error_message may vary depending on the camera models.

- OK
```
"error": [0, "OK"]
```
- Any
  A generic error code which can be used with any error.
```
"error": [1, "Any"]
```
- Timeout
```
"error": [2, "Timeout"]
```
- Illegal Argument
  Parameters in "params" are illegal.
```
"error": [3, "Illegal Argument"]
```
- Illegal Data Format
```
"error": [4, "Illegal Data Format"]
```
- Illegal Request
  When request body is empty, has no id or invalid id, has no method, has no parameter, or when "params" is not an array.
```
"error": [5, "Illegal Request"]
```
- Illegal Response
```
"error": [6, "Illegal Response"]
```
- Illegal State
```
"error": [7, "Illegal State"]
```
- Illegal Type
```
"error": [8, "Illegal Type"]
```
- Index Out Of Bounds
```
"error": [9, "Index Out Of Bounds"]
```
- No Such Element
```
"error": [10, "No Such Element"]
```
- No Such Field
```
"error": [11, "No Such Field"]
```
- No Such Method
  For cases method is unmatched.
```
"error": [12, "No Such Method"]
```
- NULL Pointer
```
"error": [13, "Null Pointer"]
```
- Unsupported Version
```
"error": [14, "Unsupported Version"]
```
- Unsupported Operation
```
"error": [15, "Unsupported Operation"]
```
- Shooting fail
```
"error": [40400, "Shooting fail"]
```
- Camera Not Ready
```
"error": [40401, "Camera Not Ready"]
```
- Already Running Polling API

```
"error": [40402, "Already Running Polling Api"]
```

- Still Capturing Not Finished

```
"error": [40403, "Still Capturing Not Finished"]
```

In case HTTP status code is other than 200 OK, the error_code will be same as HTTP status code.

Here, major status codes are listed up.

- 401 Unauthorized

```
"error": [401, "Unauthorized"]
```

- 403 Forbidden

```
"error": [403, "Forbidden"]
```

- 404 Not Found

```
"error": [404, "Not Found"]
```

- 406 Not Acceptable

```
"error": [406, "Not Acceptable"]
```

- 413 Request Entity Too Large

```
"error": [413, "Request Entity Too Large"]
```

- 414 Request-URI Too Long

```
"error": [414, "Request-URI Too Long"]
```

- 501 Not Implemented

```
"error": [501, "Not Implemented"]
```

- 503 Service Unavailable

```
"error": [503, "Service Unavailable"]
```

JSON data types

For basic information about JSON data types, refer to RFC 4627. Camera Remote API adapts some extensions to keep APIs simple and easy-to-use.

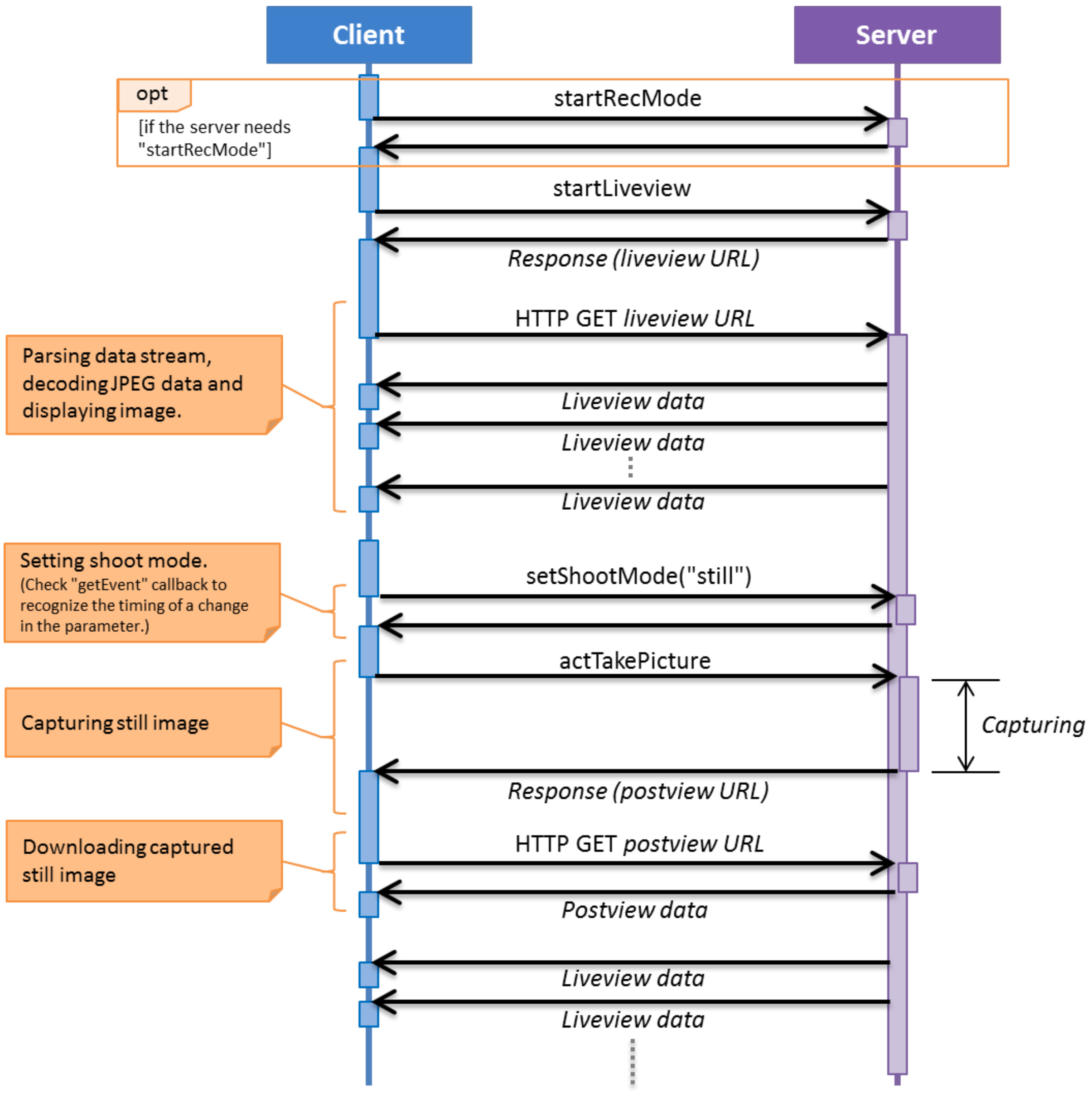
Camera Remote API defines custom data types as below.
[boolean]: true or false value.
[integer]: Integer number, ranging from -2147483648 to 2147483647.
[double]: Double number, ranging from 2.2250738585072014e-308 to 1.7976931348623157e+308.
These three data types, plus [string], are primitive data types.

Camera Remote API also defines custom array types which only contains each of the above data types.
[boolean-array]: Multiple true or false values are stored in an array.
[integer-array] : Multiple integer numbers are stored in an array.
[double-array] : Multiple double numbers are stored in an array.
[string-array] : Multiple string data are stored in an array.

Sample Sequence

Here are sample sequences for some use cases.

Displaying liveview and capturing picture

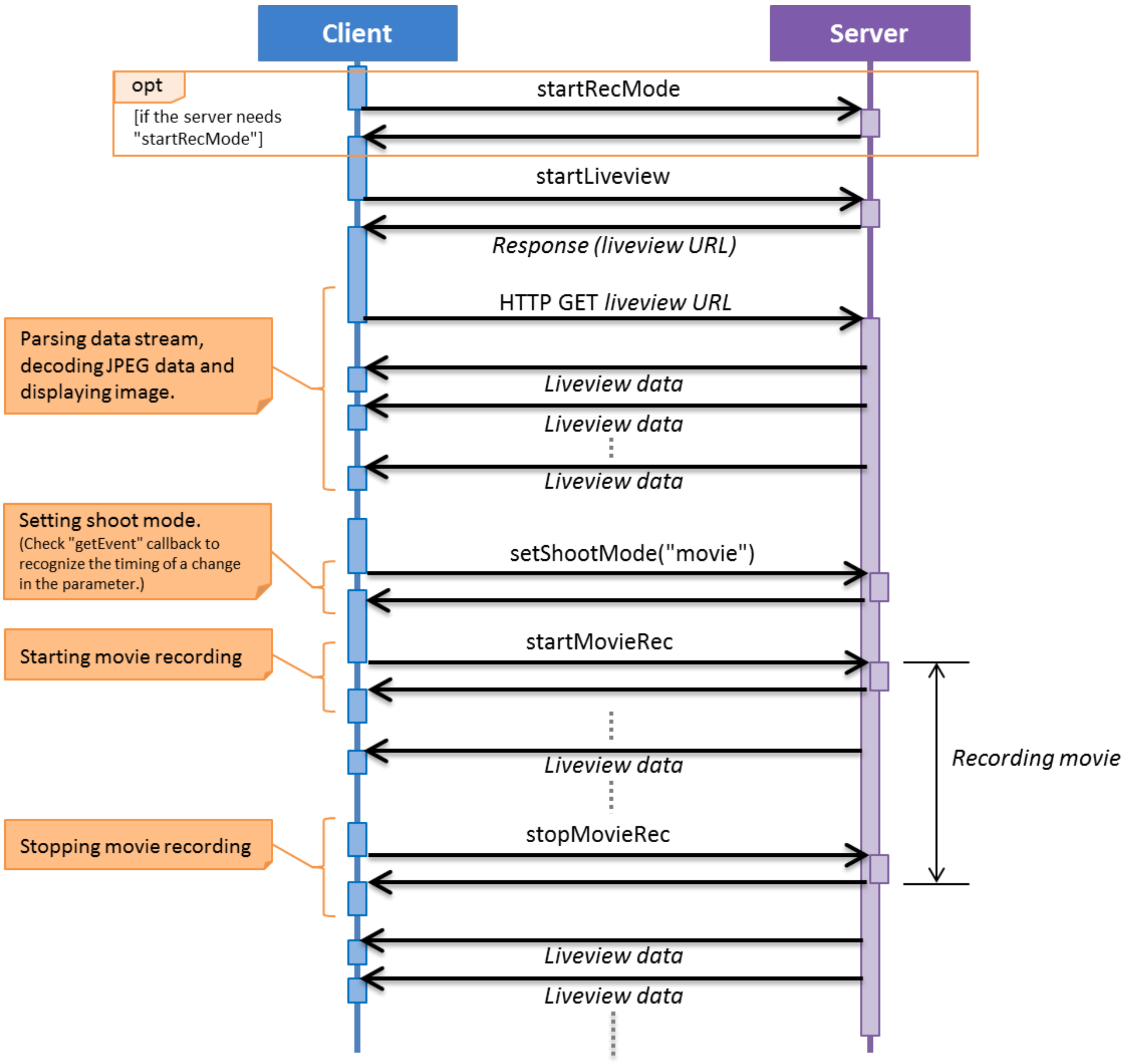


Some camera models need "startRecMode" API call before accessing camera remote shooting functions. The client must check if the server needs "startRecMode" API call. The check can be done by checking the availability of "startRecMode" API in "getAvailableApiList" or "getMethodTypes" callback.

To start liveview, the client should call "startLiveview" API and get liveview image data via the liveview URL in the response. The liveview data is kind of continuous images and the client can parse this data stream and decode each JPEG data. For more information about liveview data, see Liveview data format.

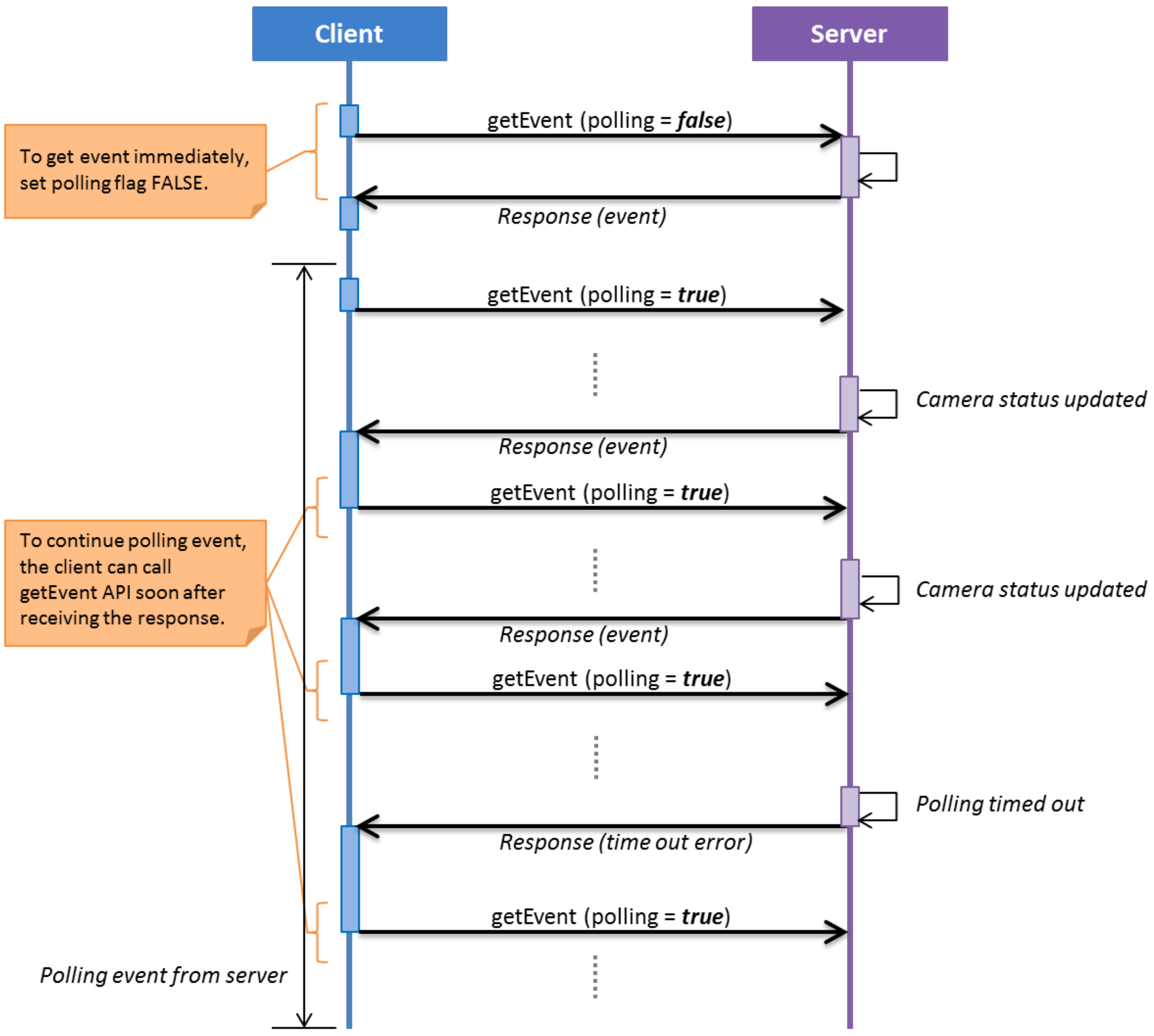
To capture still image, the client should call "actTakePicture" API in "still" shoot mode. The camera will provide the URL of postview in the response after capturing and the client can use it to display the postview image on the display and save it as the captured picture.

Recording movie



To record movie, the client should change shoot mode to "movie". The client should call "startMovieRec" API to start recording movie and call "stopMovieRec" API to stop.

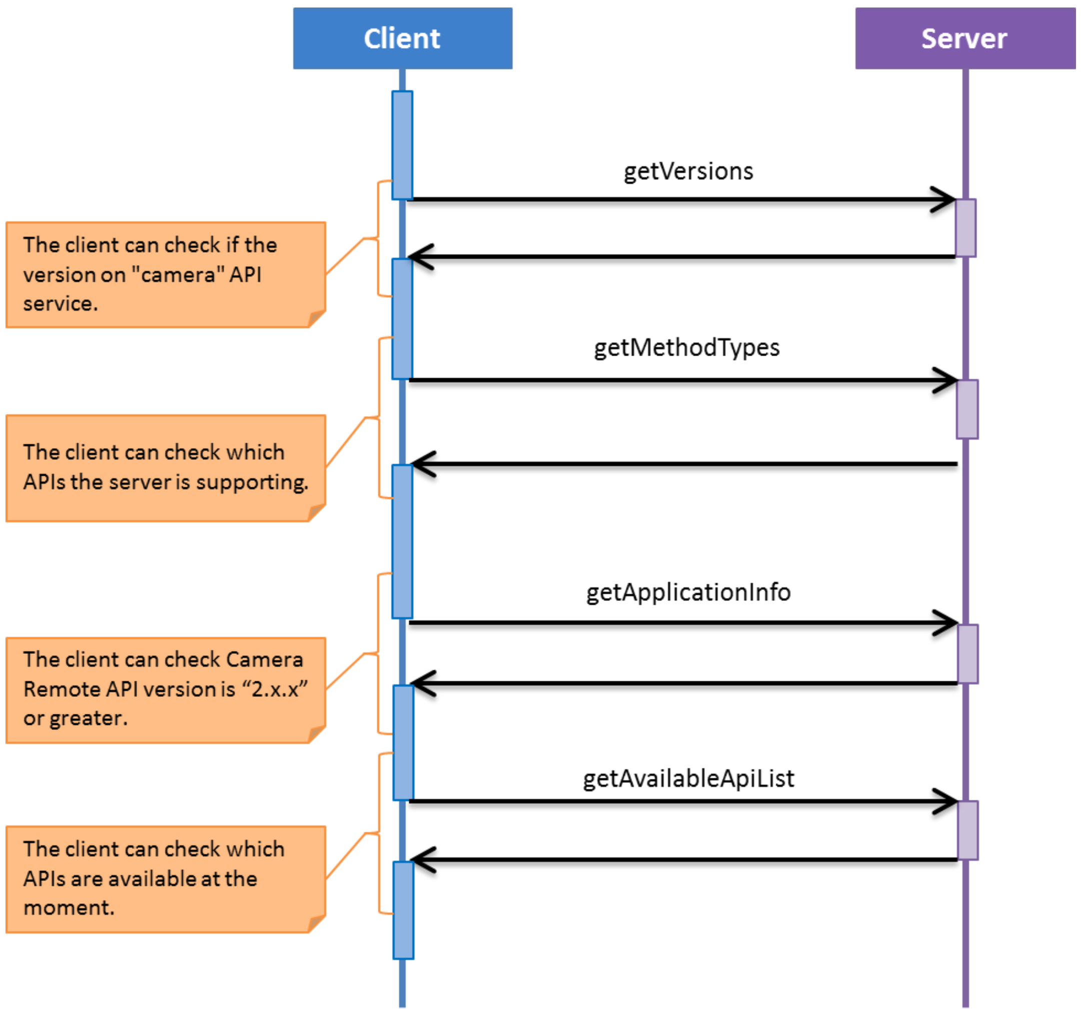
Getting event


Client
Server
To get event immediately, set polling flag FALSE.
getEvent (polling = *false*)
Response (event)
getEvent (polling = *true*)
Camera status updated
Response (event)
getEvent (polling = *true*)
To continue polling event, the client can call getEvent API soon after receiving the response.
Camera status updated
Response (event)
getEvent (polling = *true*)
Polling timed out
Response (time out error)
getEvent (polling = *true*)
Polling event from server

The purpose of "getEvent" API is sending the event from the server actively. The client can recognize the current camera status. For example, when the client calls "Zoom" API, the server will send the event including the information about zoom position.

If "getEvent" is executed with "polling=false", the server replies immediately. If "getEvent" is executed with "polling=true", the server will response when one of server parameter is updated, or timed out.

Checking API version, supported APIs and available APIs


Client
Server
getVersions
getMethodTypes
getApplicationInfo
getAvailableApiList
The client can check if the version on "camera" API service.
The client can check which APIs the server is supporting.
The client can check Camera Remote API version is “2.x.x” or greater.
The client can check which APIs are available at the moment.

The "getApplicationInfo" API provides the Camera Remote API version which the camera supports. The client should check whether the API version is "2.x.x" ("2.0.0" or greater) to confirm the server function compatibility with this document.

The functions of camera vary by the model. Therefore the APIs that the camera supports vary by the model. The camera provides its supported API list via "getMethodTypes" API.

In addition, the camera status can be changed by user operations and calling APIs. Available APIs in the camera will be changed by camera status. The camera also provides available API list via "getAvailableApiList" API or "availableApiList" object of "getEvent" API callback. For more information, please see API specification.

More information

- IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels
  http://www.ietf.org/rfc/rfc2119.txt
- IETF RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1
  http://www.ietf.org/rfc/rfc2616.txt
- IETF RFC 4627, The application/json Media Type for JavaScript Object Notation (JSON)
  http://www.ietf.org/rfc/rfc4627.txt
- JSON-RPC
  http://json-rpc.org/

For information regarding the latest Camera Remote API SDK updates, go to Developer World available at http://developer.sony.com

Trademarks and acknowledgements