TN029 Controlling Client Devices via JSON Interface

Version 4.3

Tripleplay Services Ltd.
Rapier House
40-46 Lamb's Conduit Street
London
WC1N 3LJ

www.tripleplay.tv

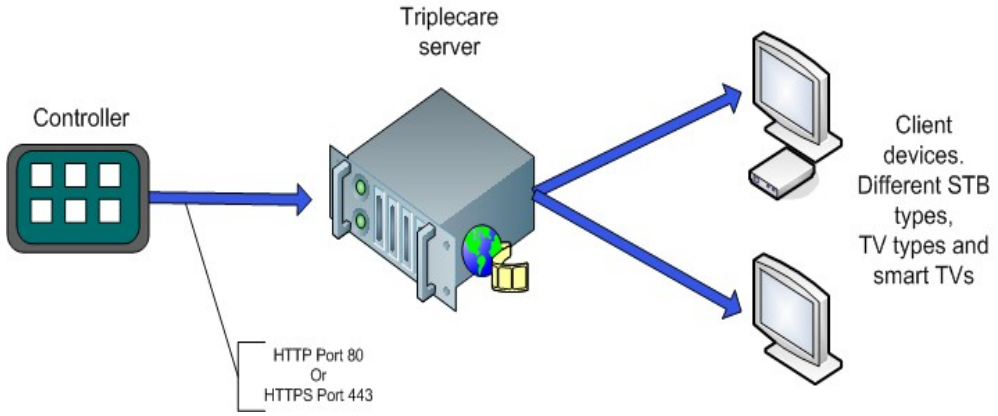
Table of Contents

1 Introduction

Tripleplay provides a JSON/XMLRPC handler which provides a means for external controllers to manage aspects of client devices connected to a Tripleplay IPTV system. The control is device agnostic meaning a single external request can control many different device types.

This document describes the interface which is applicable to the Bart 3.0 or later releases.[1]

The general architecture of using the handler is shown below. All requests are made to the Triplecare server using the HTTP protocol using port 80, or HTTPS using port 443. The Triplecare server in turn manages the client devices using appropriate communication protocols. This arrangement removes the need for the controllers to know the details of each client device, instead the Triplecare server performs all the translations and routing of messages as needed.



The controller can be any device capable of being programmed to interact with an end user and of communicating with the Triplecare server. Typically a Crestron, AMX or other bespoke control device such as a tablet or touch screen computer.

The following are a selection of commonly used examples using the Triplecare XmlRpc API which will allow external control applications to manage commonly used functions. If additional functions are needed please contact your Tripleplay representative for details, referring to the "T*ripleCare Web Service API 5.0*" or later document.

In the following sections, client ID refers to the client device which may be a Set Top Box or an integrated device such as a Smart TV.

[1]Note, this version uses the `JsonRpcHandler` call which is now the preferred interface. The older `JsonXmlRpcBridge` call is still supported for backwards compatibility but should not be used for new systems.

1.1 Identifying client devices

Commands are sent to a specific client, or a group of clients, identified in this document as the ClientID. There are a number of ways of identifying clients shown below:

Parameter format	Description.
"12"	The client referenced by ID 12[2] This format is retained for backward compatibility.
"client://12"	The client referenced by ID 12 This format supersedes the integer format above.
"clientgroup://5"	The clients in the client group referenced by client group ID 5
"location://Room 101"	The clients in Room 101. The location value is managed via the TripleCare client management pages.
"serial://KE0914D0000205"	The client with serial KE0914D0000205.
"mac://00:02:02:50:91:8f"	The client with MAC Address 00:02:02:50:91:8f
"auxiliaryId://101.1"	The client with the Auxiliary ID 101.1 This is a value that allows the customer to use their own ID (e.g. inventory ID) for the client. This will allow them to define an ID before the device is commissioned so the AMX controller can be configured ready for the commissioned client. The only way to update the Auxiliary ID is via TripleCare.

The screen-shot below shows the Triplecare Client Management screen and the details available for the clients.

Address	MAC Address	Serial Number	Location	Auxiliary Id	Locale	Last Modified	SW Version	SecureMedia	Type	Model / Version
192.168.244.101	f8:77:b8:db:ec:cb	CPCJS3W74TKFA	Block A	Asset 2030	English (United Kingdom)	2016-12-01 16:13:38	T-GFSLE5AKUC-1007.3	N/A	SamsungLFDTV	DB10E / unknown
192.168.244.102	00:02:02:5e:8c:08	KE4615D0070414	Room 101	Asset 2031	English (United Kingdom)	2016-11-24 08:50:10	4.5.0-Ax5x-opera12-tripleplay-64263.B-DI-100		Amino	A150 / 0

[2]The ID is an internal reference ID. To obtain the ID number, browse to the "Client Management → Set Top Boxes" page in the Triplecare application. Hover over the "View" button. The browser will display the URL. The number following view= is the client ID number. This number will remain unchanged for the time that client is registered with the system.

1.2 Connecting to the handler

Both HTTP (port 80) and HTTPS (port 443) can be used to communicate with the handler. In all the examples below HTTP is shown.

1.3 URL Encoding

Throughout this document the URL's specified use quotation marks (e.g. "). Some software tools (like Outlook, Word) like to automatically change the quotation marks from the standard ASCII quotation mark to 'smart quotes' which use the unicode left and right quotation marks to improve readability. Compare the quotation marks used in the following URL examples:

With unicode left and right quotes:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={“jsonrpc”:“2.0”,“method”:“Reboot”,“params”:[1]}
```

With the ASCII quotes:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"Reboot","params":[1]}
```

It is important that whenever you are using quotation marks in URLs that you use the ASCII quotation mark character, otherwise the JSON RPC handler will complain that the JSON received is invalid. To ensure that the command will always work you can replace the quotation mark with the HTTP escape code %22 like in the following URL:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?call={%22jsonrpc
%22:%222.0%22,%22method%22:%22Reboot%22,%22params%22:[1]}
```

Whilst this ensures that the URL wont be modified by email clients it does affect the readability of the URL.

1.4 Call format

All of the functions are in the format of a function call to the JSON/XMLRPC handler with a method and the necessary parameters.

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"methodA","params":[x,y,z]}
```

For example, the following call would control the Client with ID = 1 and perform a Channel Up command:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0", "method":"ChannelUp","params":[1]}
```

In all of the cases, the calls should return "true" if everything was fine, otherwise it will return a faultCode. For example the return from a successful call would be coded as:

```
{"jsonrpc":"2.0","id":null,"result":true}
```

If an error occurs in the function then the call would return an error code and an error string. For example the result of trying to set the audio volume against a non registered client:

```
{"jsonrpc":"2.0","id":null,"error":{"code":300,"message":"Client Id
not registered"}}
```

1.5 Notes on usage

Controllers should not call the methods more often than needed to perform an action.

Methods that return potentially a lot of data such as the GetAllServices should only be called once at initialisation and data cached locally in the controller.

Controllers should always check the return code from every method and take appropriate action.

2 IPTV Channel Control Functions

This set of functions control the IPTV TV channels played on clients. The channels must have been configured in Triplecare before these functions can be used.

2.1 Channel Up

This function will increment the current channel number the client is currently playing.

The Channel Up method call has the following parameters:

- **Method**: ChannelUp
- **Parameters**: Client ID

For example, this would control the client with ID = 1 and perform a Channel Up command:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0", "method":"ChannelUp","params":[1]}
```

2.2 Channel Down

This function will decrement the current channel number the client is currently playing.

The Channel Down method has the following parameters:

- **Method**: ChannelDown
- **Parameters**: Client ID

For example, this would control the client with ID = 1 and perform a Channel Down command

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0", "method":"ChannelDown","params":[1]}
```

2.3 Channel Select

This function will set the current channel number the client is currently playing. The channel ID should be obtained from Triplecare. To aid in this the function in section 8.1 can be used to find the ChannelIDs together with the channel names.

The Channel Select method has the following parameters:

- **Method**: SelectChannel
- **Parameters**: Client ID, ChannelID

For example, this would control the client with ID = 1 and change to ChannelID 2

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"SelectChannel","params":[1,2]}
```

Note, the ChannelID is not the same as the channel number that may be presented to a user, for example in an EPG or channel list.

3 Audio Control Functions

This set of functions controls the audio volume of the client devices.

3.1 Set the volume

This function will set the audio volume on the client device.

- **Method**: SetVolume
- **Parameters**: Client ID, Volume level. The volume is a value between 0 and 100.

For example, the following will set the volume on client with ID = 1 to 50%.

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"SetVolume","params":[1,50]}
```

3.2 Set Mute

This function will set the mute the audio volume on the client device without changing its volume setting.

- **Method**: SetMute
- **Parameters**: Client ID, Mute state (true or false).

For example, the following will mute the audio on client with ID = 1

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"SetMute","params":[1,true]}
```

3.3 Get Volume State (supported in Caveman 1.0 and later)

This function will get the current audio volume and mute state that the client device currently uses. This is the state of the client device, not any connected display devices.

- **Method**: GetAudioState
- **Parameters**: Client ID

For example, the following will get the audio state for device with ID = 1.

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"GetAudioState","params":[1]}
```

The state is returned as a JSON document:

TBD.

3.4 Get Mute Status (supported in Caveman 1.0 and later)

This can be obtained using the GetAudioState call described above.

4 Client Control Functions

These functions control the client state.

4.1 Reboot the client

- **Method**: Reboot.
- **Parameters**: Client ID.

For example to reboot an client client with Id = 1:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"Reboot","params":[1]}
```

4.2 Emulate an RCU KeyPress

This function tells the client to behave as if an RCU command had been received.

- **Method**: HandleKeyPress.
- **Parameters**: Client ID, Key

For example to emulate the UP key press with ID=1:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"HandleKeyPress","params":[1, "Up"]}
```

There are some common key values used for all client types, these provide all the features that are common to all the RCU units. The values are case insensitive.

Number0
Number1
Number2
Number3
Number4
Number5
Number6
Number7
Number8
Number9
ChannelUp
ChannelDown
PageUp
PageDown
Up
Down
Left
Right
Red
Green
Yellow
Blue

OK
Home
Back
Power
VolumeUp
VolumeDown
Mute
Source
Rewind
Play
Pause
Forward
Stop
Record
Guide
TV
Info
Movies
PVR
Music
Titles

- **TV Input Source Functions**

4.3 Select the Input

- **Method**: SelectTVInput.
- **Parameters**: Client ID, Input Device ID

For example to select the HDMI2 input for a client with ID = 1:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"SelectTVInput","params":[1,154]}
```

Note, the input devices are client specific, but the following set is typical. The parameter passed to SelectTVInput, is the numeric value, not the text string. E.G. parameter is 154 to select HDMI2.

Input Device	Input Device ID
VGA	103
HDMI	101
HDMI1	153
HDMI2	154
HDMI3	155
HDMI4	156
DVI	102
Display Port	139
PC	160
MEDIA	158

5 TV Power State Functions

These functions control the power state of the display devices. For these functions to work, the TV must be configured in Triplecare with the correct TV configuration, and where needed appropriate physical connectivity must be present between the STB and the TV, for example using an RS232 control.

These functions will return an error code if the configuration is missing in Triplecare.

5.1 Power on the TV

This method will power on the TV where possible.

- **Method**: PowerOnTv
- **Parameters**: Client ID

For example, the following will power on the TV with client ID = 1

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"PowerOnTv","params":[1]}
```

5.2 GetPowerStatus (supported in Caveman 1.0 and later)

TBD

5.3 Power off the TV

This method will power off the TV where possible.

- Method: PowerOffTv
- Parameters: Client ID

For example, the following will power off the TV with client ID = 1

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"PowerOffTv","params":[1]}
```

6 TV Volume Function

6.1 Set the TV Volume

- **Method**: SetTVVolume
- **Parameters**: Client ID, Volume level. The volume is a value between 0 and 100.

For example to set the volume to 30% for a client with ID = 1:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"SetTVVolume","params":[1,30]}
```

7 VoD Functions

7.1 Play a VoD Asset

- **Method**: ChangePortalPage
- **Parameters**: Client ID, action, asset.

For example to play an existing VoD asset use this:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"ChangePortalPage","params":
[1,"watch_video",
{"vodItem":"Indiana_Jones_and_the_Kingdom_of_the_Crystal_Skull"}]}
```

Would play the asset "Indiana_Jones_and_the_Kingdom_of_the_Crystal_Skull" on the client with Id=1

8 Service Functions

8.1 Get a list of all IPTV services configured

- **Method**: GetAllServices
- **Parameters**: Client ID. Note if a client ID of -1 is provided, all services will be returned.

For example to retrieve a list of all services configured against a client with an ID = 1 in Triplecare, use the following call:

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"GetAllServices","params":[1]}
```

This will return all configured services for that client as an json document. This contains all the configured details including he multicast address, port number and channel number. An example response is shown below. Additional fields may be present and should be ignored. (Note, this has been formatted for readability.)

```
{"jsonrpc":"2.0","id":null,"result":
  [{"id":1,
       "channelNumber":1,
       "name":"BBC ONE West",
       "typeSpecificData":
            {
            "ipAddress":"239.35.4.0",
            "port":1234,
            "videoCodec":"mpeg2"
            },
  }]
}
```

To get a list of all services configured for all clients set the client ID = -1 as in the following example.

```
http://<serverIP>/triplecare/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"GetAllServices","params":[-1]}
```

9 Recording Functions

The TripleTV+ recording functions can be controlled using the JSON RPC calls to start, query and stop recordings. Stopped recordings will be transferred to VoD if the 'Transfer to VoD' flag is ticked in TripleTV+->Configuration.

These functions create ad-hoc recordings which start as soon as the command is received and continue until the Stop command is received.

Recording must be licensed on the server.

9.1 Start a recording

- **Method**: RecordChannel
- **Parameters**: User, {ChannelName, RecordingName, RepeatSchedule, RepeatDays}

For example to record a channel called '*Entertainment I*' to a recoding called '*Jtest*', with no repeat schedule, use the following call:

```
http://<serverIP>/tripleshift/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"RecordChannel","params":[ "admin",
{"ChannelName": "Entertainment%20I",
"RecordingName":"JTest","RepeatSchedule":false,"RepeatDays":0}]}
```

The user can be any value,

The ChannelName is the name the programme has been given in TripleCare->services.

Note: Any non-printing characters in the channel name will need to be replaced with their hex equivalents.

This will start a recording and return the status of call including the scheduled ID of the recording. Note this is not the same as the recoding ID needed by StopRecording.

```
{"jsonrpc":"2.0","id":null,"result":"S33"}
```

9.2 Get a list of recordings

After you have started a recording you can get the list of recordings. This will allow you to determine the recording ID, which is needed to stop the recordings.

- **Method**: ViewRecordings
- **Parameters**: User, flags

Flags can be one or more of the following:

- S – include scheduled recordings
- I – include in-progress recordings
- C – include completed recordings
- F – include failed recordings
- An empty string is equivalent to requesting all types.

For example to view all recordings for user 'admin' use the following call:

```
http://<serverIP>/tripleshift/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"ViewRecordings",
"params":"admin",""]}
```

This will return a list of all recordings as a Json document. For example after starting a recording as above, a sample output would be (re-formatted for clarity).


```
"jsonrpc":"2.0",
   "id":null,
   "result":{
       "2017-02-27 12:53:39 Entertainment I 254":{
           "Id":"R254",
           "ProgrammeId":"0",
           "RecordingName":"JTest",
           "ChannelName":"Entertainment I",
           "ServiceId":"2",
           "Status":"I",
           "Size":6866,
           "Duration":240,
           "RequestedStart":"2017-02-27 12:53:39",
           "ScheduledStart":"2017-02-27 12:53:39",
           "RequestedEnd":"2017-02-27 16:53:39",
           "ScheduledEnd":"2017-02-27 16:53:39",
           "ActualStart":"2017-02-27 12:53:39",
           "ActualEnd":"2017-02-27 16:53:39",
           "LateRequestInfo":0,
           "StbRecording":false,
           "ItemCrid":"",
           "SeriesCrid":"",
           "IsSeriesRecording":false,
           "Rating":"0",
           "ServiceAccessControl":"1",
           "Description":""
       }
   }
```


9.3 Stop Recording

After you have started a recording you can get the list of recordings. This will allow you to determine the recoding ID, which is needed to stop the recordings.

- **Method**: StopRecording
- **Parameters**: User, RecordingID as returned in ViewRecordings.

The user must be the same as the user that started the recording. For example to stop the recording started above:

```
http://<serverIP>/tripleshift/JsonRpcHandler.php?
call={"jsonrpc":"2.0","method":"StopRecording",
"params":["admin","R254"]}
```

The return will be the status of the stop request. A successful stop will return *true*.

```
{"jsonrpc":"2.0","id":null,"result":true}
```

The recording will be stopped, and the content transferred to VoD. The following settings are used to categorise the recording:

Name: The name of the recording specified in RecordChannel. In the above examples this is Jtest.

Family: User Recordings

Category: User Recordings

If you require other families or categories then the content will need to be edited in TripleCMS to change the meta data.

10 Query Clients List

10.1 QueryClients function

It takes between 1 and 5 parameters.

Parameter 1: Array of filter objects
e.g.

```
[
    {"field":"clientType","operator":"is","value":"STB"}
]
```

Parameter 2: Array of output options
e.g.

```
{
    "hardware":false,
    "network":true,
    "activity":false,
    "services":false,
    "configuration":false
}
```

Parameter 3: Sort Field e.g. ipAddress
Parameter 4: The maximum number of results to return
Parameter 5: The page of results to return

The data can be filtered using the following parameters and operators:

```
STB Clients
   Field: ipAddress Type: string
   Field: externalIp Type: string
   Field: macAddress Type: string
   Field: auxiliaryId Type: string
   Field: description Type: string
   Field: location Type: string
   Field: locale Type: string
   Field: dateLastUpdated Type: date
   Field: type Type: string
   Field: swVersion Type: string
   Field: serial Type: string
   Field: hwVersion Type: string
   Field: model Type: string
   Field: deploymentIndex Type: number
   Field: smVersion Type: string
```

```
PC Clients
   Field: ipAddress Type: string
   Field: externalIp Type: string
   Field: macAddress Type: string
   Field: description Type: string
   Field: location Type: string
   Field: locale Type: string
   Field: dateLastUpdated Type: date
   Field: username Type: string
   Field: operatingSystem Type: string
   Field: browser Type: string
   Field: browserVersion Type: string
   Field: videoControlVersion Type: string

Mobile Clients
   Field: ipAddress Type: string
   Field: externalIp Type: string
   Field: macAddress Type: string
   Field: description Type: string
   Field: location Type: string
   Field: locale Type: string
   Field: dateLastUpdated Type: date
   Field: username Type: string
   Field: platform Type: string
   Field: formFactor Type: string
   Field: osVersion Type: string
   Field: name Type: string
   Field: appName Type: string
   Field: appVersion Type: string

String Operators
   is
   is not
   starts with
   contains
   ends with
   does not start with
   does not contain
   does not end with

Date Operators
   on
   not on
   before
   on or before
   after
   on or after

Number Operators
   equals
   not equals
   less than
   less than or equal
   greater than
   greater than or equal
```

Adding Multiple Filters

Below are examples of including multiple filters with OR's or AND's...

```
var filters = [
    {"field":"ipAddress","operator":"is","value":"192.168.242.101"},

{"logical":"OR","field":"ipAddress","operator":"is","value":"192.168.242.
103"}
]

var filters = [
    {"field":"type","operator":"is","value":"Amino"},
    {"logical":"AND","field":"model","operator":"is","value":"A150"}
]
```

Examples:

```
https://<server-address>/triplecare/JsonRpcHandler.php?
call={%22jsonrpc%22:2.0,%22method%22:%22QueryClients%22,%22params%22:
[[{%22field%22:%22clientType%22,%22operator%22:%22is%22,%22value
%22:%22STB%22}],{%22hardware%22:true,%22network%22:true,%22activity
%22:false,%22services%22:false,%22configuration%22:false},%22ipAddress
%22,20,1]}
```

Replace the <server-address> with the IP or the domain of the server.

Sample reply below:


```
{
  "jsonrpc": "2.0",
  "id": null,
  "result": {
    "clients": [
      {
        "clientId": "16",
        "locale": "en_GB",
        "location": "Tripleplay",
        "auxiliaryId": "",
        "description": "",
        "type": "STB",
        "typeDescription": "MvpStb MvpStb Client",
        "hardware": {
          "type": "MvpStb",
          "softwareVersion": "1.2",
          "serialNumber": "1",
          "hardwareVersion": "0",
          "model": "MvpStb"
        },
        "network": {
          "ip": "192.168.245.19",
          "mac": "80:ee:73:b9:b6:ef",
          "homepage": false,
          "dhcpSubnet": "Dynamic"
        }
```

```
      },
      {
        "clientId": "3",
        "locale": "en_GB",
        "location": "XT1144",
        "auxiliaryId": "",
        "description": "",
        "type": "STB",
        "typeDescription": "BrightSign XT1144 Client",
        "hardware": {
          "type": "BrightSign",
          "softwareVersion": "7.1.65",
          "serialNumber": "D7E87U002254",
          "hardwareVersion": "0",
          "model": "XT1144"
        },
        "network": {
          "ip": "192.168.245.30",
          "mac": "90:ac:3f:11:11:a8",
          "homepage": "http://192.168.245.1/portal/",
          "dhcpSubnet": "Dynamic"
        }
      },
      {
        "clientId": "7",
        "locale": "en_GB",
        "location": "H150",
        "auxiliaryId": "",
        "description": "",
        "type": "STB",
        "typeDescription": "Amino H150 Client",
        "hardware": {
          "type": "Amino",
          "softwareVersion": "5.7.7-Ax5x-opera12-tripleplay-82278.T-
DI-100",
          "serialNumber": "KN2118D0041868",
          "hardwareVersion": "0",
          "model": "H150"
        },
        "network": {
          "ip": "192.168.245.33",
          "mac": "00:02:02:6b:15:0c",
          "homepage": "http://192.168.245.1/portal/",
          "dhcpSubnet": "Dynamic"
        }
      },
      {
        "clientId": "6",
        "locale": "en_GB",
        "location": "SP1",
        "auxiliaryId": "",
        "description": "",
        "type": "STB",
        "typeDescription": "TpsBrightSign TPS-SP1-4 Client",
        "hardware": {
          "type": "TpsBrightSign",
          "softwareVersion": "8.0.69",
          "serialNumber": "51D99S001778",
          "hardwareVersion": "0",
          "model": "TPS-SP1-4"
```

```
        },
        "network": {
          "ip": "192.168.245.35",
          "mac": "90:ac:3f:18:55:b0",
          "homepage": "http://192.168.245.1/portal/",
          "dhcpSubnet": "Dynamic"
        }
      }
    ],
    "total": "4",
    "limit": 20,
    "page": 1
  }
}
```


For Page 2 results, call:

https://<server-address>/triplecare/JsonRpcHandler.php?call={%22jsonrpc%22:2.0,%22method%22:%22QueryClients%22,%22params%22:[[{%22field%22:%22clientType%22,%22operator%22:%22is%22,%22value%22:%22STB%22}],{%22hardware%22:true,%22network%22:true,%22activity%22:false,%22services%22:false,%22configuration%22:false},%22ipAddress%22,20,2]}

END OF DOCUMENT