# External Control Protocol (JSON) Reference Manual

020-102837-04

# Terra

CHRISTIE

## Revision History

| Date | Version | Revision | Description |
|---|---|---|---|
| 04/30/18 | 1 | 1 | First release. |
| 9/12/18 | 2 | 1 | R1.0.4. Changed the RS-232 baud rate to 9600. |
| 10/25/18 | 3 | 1 | Added the following commands:<br>GetLayoutByName, SwitchVideo, SwitchAudio |
| 11/1/18 | 4 | 1 | Added the |

# NOTICES



### GENERAL

Every effort has been made to ensure accuracy, however in some cases changes in the products or availability could occur which may not be reflected in this document. Christie reserves the right to make changes to specifications at any time without notice. Performance specifications are typical, but may vary depending on conditions beyond Christie's control such as maintenance of the product in proper working conditions. Performance specifications are based on information available at the time of printing. Christie makes no warranty of any kind with regard to this material, including, but not limited to, implied warranties of fitness for a particular purpose. Christie will not be liable for errors contained herein or for incidental or consequential damages in connection with the performance or use of this material. Canadian manufacturing facility is ISO 9001 and 14001 certified.

# Content

# Terra API

This document describes the JSON interface control protocol commands for remote access to the Terra system. This protocol is based on the JSON-RPC 2.0 specification.

# Reference and Related Documents

Access the latest documentation from the Christie website at http://bit.ly/TerraDownloads

Additional information is available in the following documents:

- Terra Installation and Setup Guide for Controlled Systems (020-102804-*nn*)
- Terra Installation and Setup Guide for Transmitters and the Receivers (020-102814-*nn*)
- Terra Product Safety Guide (020-102786-*nn*)
- Terra User Manual (020-102838-*nn*)
- Terra XY Switcher User Manual (020-102883-*nn)*
- Terra XY Switcher API Reference Manual (020-102884-*nn*)

# Transport and Framing

There are several transport methods allowed for this protocol. Message framing varies by transport protocol. All commands should respond within 15 seconds (round trip) unless otherwise specified.

Only one RS-232 connection is supported at one time.

# TCP

The baseline communication method for JSON-RPC will be TCP. Each session will consist of one TCP session.

Port 23456 is the TCP/IP port used for JSON communications.

Messages must be framed using the CRLF characters. That is, messages must be separated by the {0x0D, 0x0A} characters. Any data before the two null characters that is not valid JSON is an invalid message and will generate an error messages.

Follow all commands with a Carriage Return followed by a New Line.

Asynchronous notifications may be supported by the server. Therefore, if a client sends overlapping requests, the server may return the replies out-of-order. The client must use the JSON-RPC `id` value to distinguish replies. Server notifications are inserted into the connection that registered for them, therefore a client waiting for a reply may receive other messages before the appropriate reply. A well-behaved client should not open more than 2 TCP connections.

Flow control is handled by TCP. When the TCP stream is broken, any pending registrations/notifications are cleared.

# Serial Port (COM1) (RS-232)

JSON RPC can be used over a serial port link via the COM1 port on the controller. The communication parameters must be set as follows:

| | |
|---|---|
| Baud Rate | 9600 |
| Parity | None |
| StopBits | 1 |
| DataBits | 8 |
| Handshake | None |

Other than the hardware link, all messaging is identical to that of TCP.

The Serial RS232 or Network & RS232 option must be selected in the API Connection page in the Web Manager Global Settings page to enable this method of communication.

# Standards and Conventions

1. RFC 4627, "The application/json Media Type for JavaScript Object Notation (JSON)", http://www.ietf.org/rfc/rfc4627
2. JSON-RPC 2.0 Specification, http://www.jsonrpc.org/specification
3. M Series Serial Commands Technical Reference 020-100224-*nn*

RFC 4627 defines the underlying data encoding format, while the JSON-RPC 2.0 specification outlines the message structure, message semantics, and request, response protocol.

The Terra protocol supports all features of JSON-RPC 2.0 including notifications and batch messages.

This document should not be read without reference to the above to specifications. In particular, the nuances of the JSON-RPC semantics and of JSON encoding are not described in this document.

# Command and Event Names

All commands and event names are all lowercase letters. For ease of reading, this document uses initial capital letters in the document headings.

# Parameters

For any message that specifies no parameters, the `params` member may either be omitted (as per JSON-RPC 2.0 spec), or transmitted as an empty array [] or object {}.

# Batch Messages

Batch messages are supported as described by the JSON-RPC 2.0 specification with the caveat that they will be executed in order. That is, the second method in a batch request may rely on the first. Note: there may be some commands whose response does not necessarily indicate completion.

# Languages

When the language for a Terra controller has been set, all labels/names are localized using UTF-8 encoding.

# Reply and Acknowledgement

Reply and acknowledgement follow the JSON-RPC 2.0 standard.

Notifications have no `id` value and are used by the Terra controller for notifications.

Requests must have an `id` value, this value may be any JSON primitive value, though it should normally be a unique number within the range of a signed int32. JSON requests may be used for either a request/query or for a command, where an acknowledgement is desired.

JSON-RPC 2.0 requests are nominally asynchronous, though a client may employ synchronous communication by not sending a subsequent request until previous replies have been received.

# Error Messages

Protocol-level errors must comply with the JSON-RPC 2.0 standard error codes and formats.

Generic errors such as invalid (or missing) parameters, etc. are not explicitly specified in this document. However, runtime errors are occasionally spelled out, where the specific message may be of importance.

| code | message | meaning |
|---|---|---|
| -32700 | Parse error | Invalid JSON was received by the server.<br>An error occurred on the server while parsing the JSON text. |
| -32600 | Invalid request | The JSON sent is not a valid Request object. |
| -32601 | Method not found | The method does not exist / is not available. |
| -32602 | Invalid params | Invalid method parameter(s). |
| -32603 | Internal error | Internal JSON-RPC error. |

# Generic Examples

Most methods described in this document include examples, using the following notation:

To illustrate a particular client request and server response:

Request

```
{"id": 1, "jsonrpc": "2.0", "method": "abc", "params": {}}
```

Response

```
{"id": 1, "jsonrpc": "2.0", "result": 123}
```

# Asynchronous Notifications (Events)

The Terra protocol is designed to use asynchronous messaging where appropriate. Clients can register and unregister for various notification events, which are messages sent asynchronously from the server, typically upon a state change in the server.

Each namespace contains `addListener` and `removeListener` methods, and often it is possible to register for multiple (related) methods in a single call. Unless noted, the default behavior of the server is to immediately send the initial state in the form of an asynchronous notification message.

If `addListener` has been called more times than its corresponding `removeListener` method, notifications will continue. This implies that the server will count registration instances for each client.

# Events

This section describe the different types of notification events that can occur for the system or system components. Events are sent with the "method" property stating the type of event. The parameters specify the entity that was impacted by the event.

After receiving the ID from an event, use the appropriate API command to get the updated details for the ID.

# Summary of Events

The following is a summary of events and a description of when they occur:

| Event Type (method) | Description | Parameters (params) |
|---|---|---|
| deviceadded | Device was added. | id |
| devicedeleted | Device was deleted and its IP address was released. | id |
| deviceupdated | Device was updated. | id |
| displayarrayadded | Display array was added. | id |
| displayarraydeleted | Display array was deleted. | id |
| displayarrayupdated | Configuration for the display array was updated. | id |
| layoutadded | New layout was added. | id |
| layoutappliedtodisplayarray | Layout was applied to specified display array. | layoutid, displayarrayid |
| layoutdeleted | Layout was deleted. | id |
| layoutupdated | Layout was updated. | id |
| systemshutdown | System controller was powered off. | |
| systemrestart | Controller was restarted. Optionally, the Transmitters and Receivers can be restarted. | |
| systemupdating | System software update is in progress. | |

# Event Response Format

{"jsonrpc": "2.0", "method": "*method*", "params": { "id": "*entity id*" }, "id": 1}

System events do not have any parameters.

# Event Response Examples

This example is a notification that the device with id DEVICE1 was added:

{"jsonrpc": "2.0", "method": "deviceadded", "params": { "id": "DEVICE1" }, "id": 1}

This example is a notification that LAYOUTWIDE was applied to the display array with the id of DISPLAYARRAY1.

{"jsonrpc": "2.0", "method": "layoutappliedtodisplayarray", "params": { "layoutid": "LAYOUTWIDE”, “displayarrayid”: “DISPLAYARRAY1” }, "id": 1}

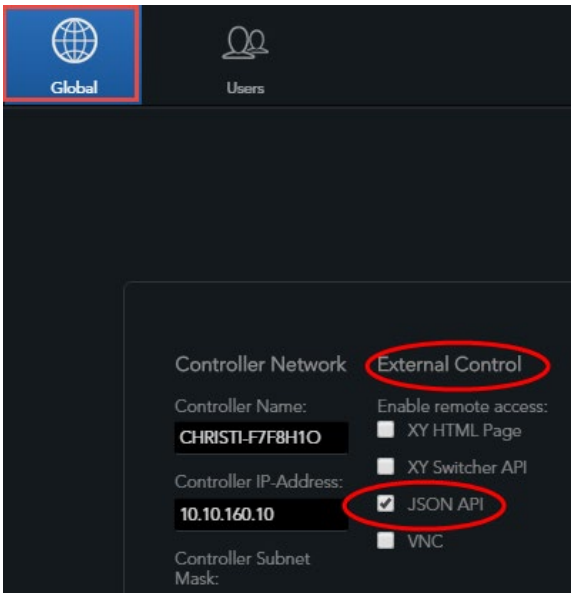
This example is a notification that a software update is in progress for the system.

{"jsonrpc": "2.0", "method": "systemupdating", "params": { }, "id": 1}

{"jsonrpc": "2.0", "method": "systemupdating", "params": { }, "id": 1}

# Enable Access and Device Configuration

Use the External Controller settings on the Terra Manager Global page to enable remote access to Terra using the JSON commands and/or Terra XY Switcher software. Access can be set up using TCP protocol or a RS-232 connection via the serial port on the Controller.




All Terra devices must be configured using Terra Manager.

CHRISTIE

# Commands

The commands are categorized by types: layout, device control and switching, device, display array, listener, and system.

# Layout Commands

This section describes the layout commands.

## ApplyLayout

Applies the specified layout to the display array is was created on.

### Request parameters

| | |
|---|---|
| layoutid | The ID of the layout you want to apply. |

### Request example

{"jsonrpc": "2.0", "method": "applylayout", "params": { "layoutid": "ENTITYID" }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

## GetLayoutByID

Gets the specified layout.

### Request parameters

| | |
|---|---|
| id | The ID of the layout you want to get. |

### Request example

{"jsonrpc": "2.0", "method": "getlayoutbyid", "params": { "id": "ENTITYID” }, "id": 1}

### Response parameters

| | |
|---|---|
| id | The ID of the layout that was retrieved. |
| name | Name of the layout that was retrieved. |

### Response example

{"jsonrpc":"2.0","result": { “id": "ENTITYID", “name”: “ENTITYNAME” },"id":1}

## GetLayoutsByRange

Returns a list of layouts within a specified range.

### Request parameters

| | |
|---|---|
| startindex | The number of the first layout to retrieve. |

| | |
|---|---|
| endindex | The number of the last layout to retrieve. |

### Request example

{"jsonrpc": "2.0", "method": "getlayoutsbyrange", "params": { "startindex": index, "endindex": index }, "id": 1}

## Response parameters

| | |
|---|---|
| ID | Layout ID that was retrieved. |
| Name | Layout Name that was retrieved. |

### Response example

{"jsonrpc":"2.0","result": { [{“id": "ENTITYID", “name”: “ENTITYNAME”}, {“id": "ENTITYID", “name”: “ENTITYNAME”}] },"id":1}

# GetLayoutCount

Returns the total number of Layouts.

## Request parameters

None.

### Request example

{"jsonrpc": "2.0", "method": "getlayoutcount", "params": { }, "id": 1}

## Response parameters

None.

### Response example

{"jsonrpc":"2.0","result":NUMBER_OF_LAYOUTS,"id":1}

# GetLayoutByName

Returns a list of layouts by name.

## Request parameters

| | |
|---|---|
| name | The name of the layout you want to get. |

### Request example

{"jsonrpc": "2.0", "method": "getlayoutbyname", "params": { "name": "LAYOUTNAME }, "id": 1}

## Response parameters

| | |
|---|---|
| id | The system id for the layout. |
| name | The name of the layout. |

### Response example

{"jsonrpc":"2.0","result": { “id": "ENTITYID", “name”: “ENTITYNAME” },"id":1}

CHRISTIE

# Device Control Commands

This section describes the device control commands.

## GenericVideoAudioSwitch

Switches to the TXs video and audio.

### Request parameters

| | |
|---|---|
| fromdeviceid | The source device ID. |
| todeviceid | The target device ID. |

### Request example

{"jsonrpc": "2.0", "method": "genericvideoaudioswitch", "params": { "fromdevice": "DEVICEID", "todevice": "DEVICEID" }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

## ApplyVideoSource

Switches a video source of a window in a layout.

### Request parameters

| | |
|---|---|
| displayarrayid | ID of the Display Array. |
| layoutid | ID of the layout. |
| windowid | ID of the window in the Display Array. |
| sourceid | ID of the source. |

### Request example

{"jsonrpc": "2.0", "method": "applyvideosource", "params": { "displayarrayid": "ENTITYID", "layoutid": "ENTITYID", "windowid": "ENTITYID", "sourceid": "ENTITYID" }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

## SendUART

Sending UART data to one or more devices.

- If sending to one device, specify the deviceid. When deviceid is specified, the targetgroup should not be specified and is ignored.
- If sending to more than one device, use one of the targetgroup parameters (NONE, ALL, ALL_TX, ALL_RX).

## Request parameters

None.

### Request example

{"jsonrpc": "2.0", "method": "senduart", "params": { "deviceid": "ENTITYID", "targetgroup": "NONE", "port": port, "data": "UART DATA" }, "id": 1}

## Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

CHRISTIE

# Switching Commands

This section describes the switching commands.

## SwitchKVM

Switching the USB signals from a Transmitters to a Receiver.

### Request parameters

| | |
|---|---|
| fromdeviceid | The source device ID. |
| todeviceid | The target device ID. |

#### Request example

{"jsonrpc": "2.0", "method": "switchkvm", "params": { "fromdevice": "DEVICEID", "todevice": "DEVICEID" }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

#### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

## SwitchVideo

Switches the video source for a device.

### Request parameters

| | |
|---|---|
| fromdeviceid | The source device ID. |
| todeviceid | The target device ID. |

#### Request example

{"jsonrpc": "2.0", "method": "switchvideo", "params": { "fromdevice": "DEVICEID", "todevice": "DEVICEID" }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

#### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

## SwitchAudio

Switches the audio source for a device.

### Request parameters

| | |
|---|---|
| fromdeviceid | The source device ID. |
| todeviceid | The target device ID. |
| type | 0 = HDMI_AUDIO<br>1 = ANALOG |

### Request example

{"jsonrpc": "2.0", "method": "switchaudio", "params": { "fromdevice": "DEVICEID", "todevice": "DEVICEID", "type": 0/1 }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

## SwitchUART

Switching the UART (universal asynchronous receiver-transmitter) signals from a Transmitter to a Receiver.

### Request parameters

| | |
|---|---|
| fromdeviceid | The source device ID. |
| todeviceid | The target device ID. |

### Request example

{"jsonrpc": "2.0", "method": "switchuart", "params": { "fromdevice": "DEVICEID", "todevice": "DEVICEID" }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

## SwitchInputConnector

Switches the input (HDMI or DisplayPort) on a Transmitter.

### Request parameters

| | |
|---|---|
| deviceid | HDMI=0<br>DisplayPort=1 |

### Request example

{"jsonrpc": "2.0", "method": "switchinputconnector", "params": { "deviceid": "DEVICEID", "connector": 0/1 }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

# Device Commands

This section describes the device commands.

## GetDeviceByID

Gets the specified device.

### Request parameters

| | |
|---|---|
| id | The ID of the device you want to get. |

### Request example

{"jsonrpc": "2.0", "method": "getdevicebyid", "params": { "id": "ENTITYID" }, "id": 1}

### Response parameters

| | |
|---|---|
| ID | Device ID that was retrieved. |
| Name | Device Name that was retrieved. |

### Response example

{"jsonrpc":"2.0","result":{"id": ENTITY_ID, "name": ENTITY_NAME, "height": 0, "width": 0, "fps": 0},"id":1}

## GetDevicesByRange

Returns a list of devices within a specified range.

### Request parameters

| | |
|---|---|
| startindex | The number of the first device to retrieve. |
| endindex | The number of the last device to retrieve. |

### Request example

{"jsonrpc": "2.0", "method": "getdevicesbyrange", "params": { "startindex": index, "endindex": index }, "id": 1}

### Response parameters

| | |
|---|---|
| ID | Device ID that was retrieved. |
| name | Device Name that was retrieved. |

### Response example

{"jsonrpc":"2.0","result":{"id": ENTITY_ID, "name": ENTITY_NAME, "height": 0, "width": 0, "fps": 0},"id":1}

# StopDevice

Stops the process of sending (TX) or receiving (RX) video.

## Request parameters

| | |
|---|---|
| deviceid (string) | ID of the device. |

### Request example

{"jsonrpc": "2.0", "method": "stopdevice", "params": { "deviceid": "DEVICEID" }, "id": 1}

## Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

# GetDeviceCount

Returns the total number of devices.

## Request parameters

None.

### Request example

{"jsonrpc": "2.0", "method": "getdevicecount", "params": { }, "id": 1}

## Response parameters

None.

### Response example

{"jsonrpc":"2.0","result":NUMBER_OF_DEVICES,"id":1}

# Display Array Commands

This section describes the Display Array commands.

## GetDisplayArrayByID

### Request parameters

| | |
|---|---|
| id | The ID of the display array to retrieve. |

#### Request example

{"jsonrpc": "2.0", "method": "getdisplayarraybyid", "params": { "id": "ENTITYID" }, "id": 1}

### Response parameters

| | |
|---|---|
| id | The ID of the display array that was retrieved. |
| name | Name of the display array that was retrieved. |

#### Response example

{"jsonrpc":"2.0","result": { “id": "ENTITYID", “name”: “ENTITYNAME” },"id":1}

## GetDisplayArraysByRange

Returns a list of display arrays within a specified range

### Request parameters

| | |
|---|---|
| startindex | The number of the first display array to retrieve. |
| endindex | The number of the last display array to retrieve. |

#### Request example

{"jsonrpc": "2.0", "method": "getdisplayarraysbyrange", "params": { "startindex": index, "endindex": index }, "id": 1}

### Response parameters

| | |
|---|---|
| id | The ID of the display array that was retrieved. |
| name | Name of the display array that was retrieved. |

#### Response example

{"jsonrpc":"2.0","result": { [{“id": "ENTITYID", “name”: “ENTITYNAME”}, {“id": "ENTITYID", “name”: “ENTITYNAME”}] },"id":1}

# ClearDisplayArray

Clears all sources from the display array.

## Request parameters

| | |
|---|---|
| id | The ID of the display array to clear. |

### Request example

{"jsonrpc": "2.0", "method": "cleardisplayarray", "params": { "id": "ENTITYID" }, "id": 1}

## Response parameters

| | |
|---|---|
| Success | True if successful. False if Terra is unable to complete the command. |

### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

# GetDisplayArrayCount

Returns the total number of Display Arrays.

## Request parameters

None.

### Request example

{"jsonrpc": "2.0", "method": "getdisplayarraycount", "params": { }, "id": 1}

## Response parameters

None.

### Response example

{"jsonrpc":"2.0","result":NUMBER_OF_DISPLAYARRAY,"id":1}

# GetWindowsForDisplayArray

Gets a list of windows showing on a Display Array. Returns an array with Display Array ID, Layout ID, Window ID, and Source ID.

## Request parameters

None.

### Request example

{"jsonrpc": "2.0", "method": "getwindowsfordisplayarray", "params": { "id": "ENTITYID" }, "id": 1}

## Response parameters

None.

### Response example

{"jsonrpc":"2.0","result":{“success”: true, windows": [{"displayarrayid": "ENTITYID", "layoutid": "ENTITYID", "windowid": "ENTITYID", "sourceid": "ENTITYID"},{"displayarrayid": "ENTITYID", "layoutid": "ENTITYID", "windowid": "ENTITYID", "sourceid": "ENTITYID"}]},"id":1}

# Listener Commands

This section describes the Listener commands. Listener commands are used to receive notifications from the system.

## addlistener

Adds a listener for the type specified in the request. Listenertypes are one of the following:

### listenertypes

**Device**

deviceadded

devicedeleted

devicedisconnected

deviceupdated

**Display Array**

displayarrayadded

displayarraydeleted

displayarrayupdated

**Layout**

layoutadded

layoutappliedtodisplayarray

layoutdeleted

layoutupdated

**System**

systemrestart

systemshutdown

systemupdating

**User**

useradded

userdeleted

userlogin

userupdated

**User Group**

usergroupadded

usergroupdeleted

usergroupupdated

## Request parameters

| | |
|---|---|
| listenertype | The type of listener to subscribe to receive notifications. Refer to *listenertypes*, page *23*. |

### Request example

The following example format can be used for any of the listener commands by replacing the *listnertype* in the example format.

{"jsonrpc": "2.0", "method": "addlistener", "params": { "listenertype": "LISTENERTYPE” }, "id": 1}

## Response parameters

None. See *Error messages* (page *7).*

### Response example

{"jsonrpc":"2.0","result":{},"id":1}

# removelistener

This method unregisters from notifications.

## Request Parameters

| | |
|---|---|
| listenertype | The type of listener to unsubscribe. |

### Request example

The following example format can be used for any of the listener commands by replacing the *listnertype* in the example format.

{"jsonrpc": "2.0", "method": "removelistener", "params": { "listenertype": "LISTENERTYPE” }, "id": 1}

## Response parameters

None. See *Error messages* (page *7).*

## Response example

**request:**

{"jsonrpc": "2.0", "method": "removelistener", "params": { "listenertype": "LISTENERTYPE" }, "id": 1}

**response:**

{"jsonrpc":"2.0","result":{},"id":1}

# removealllisteners

The top level `removealllisteners` method completely unregisters all notifications currently registered to the client. It is recommended that this is called just before a client exits.

## Request parameters

None.

## Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

## Response example

**request:**

{"jsonrpc": "2.0", "method": "removealllisteners", "params": {}, "id": 1}

**response:**

{"jsonrpc":"2.0","result":{},"id":1}

# System Commands

This section describes the system commands. System commands do not have any parameters and the response is the success status.

## RestartSystem

Restarts the controller.

### Request parameters

None.

#### Request example

{"jsonrpc": "2.0", "method": "restartsystem", "params": {  }, "id": 1}

### Response parameters

| | |
|---|---|
| success | True if successful. False if Terra is unable to complete the command. |

#### Response example

{"jsonrpc":"2.0","result":{“success”: true},"id":1}

# Index

## Corporate offices

USA – Cypress
ph: 714-236-8610

Canada – Kitchener
ph: 519-744-8005

## Consultant offices

Italy
ph: +39 (0) 2 9902 1161

## Worldwide offices

Australia
ph: +61 (0) 7 3624 4888

Brazil
ph: +55 (11) 2548 4753

China (Beijing)
ph: +86 10 6561 0240

China (Shanghai)
ph: +86 21 6278 7708

Eastern Europe and
Russian Federation
ph: +36 (0) 1 47 48 100

France
ph: +33 (0) 1 41 21 44 04

Germany
ph: +49 2161 664540

India
ph: +91 (080) 6708 9999

Japan (Tokyo)
ph: 81 3 3599 7481

Korea (Seoul)
ph: +82 2 702 1601

Republic of South Africa
ph: +27 (0)11 510 0094

Singapore
ph: +65 6877-8737

Spain
ph: + 34 91 633 9990

United Arab Emirates
ph: +971 4 3206688

United Kingdom
ph: +44 (0) 118 977 8000