By-me VIMAR

IP Connector Protocol

Specifications

Document Revision:	2.7
Written by:	VIMAR
Date:	2023–11–15
Audience:	Confidential

dat.nguyendinh@vtd.com.vn

CONFIDENTIAL

This document is the property of VIMAR S.p.A The data contained herein, in whole or in part, may not be duplicated, used or disclosed outside the recipient for any purpose other than to conduct technical evaluation. This restriction does not limit the recipient's right to use information contained in the data if it is obtained from another source without restriction.

REVISIONS

Rev.	Date	By	Pages	Short description of modifications
2.2	2021-01-07	AI	6 9 15 16 18	Introduced association for third-party clients • New sfcategory: 'ThirdParty' • New values for 'clienttag' and 'usertype' fields in the ATTACH command: 'thirdpartyapp' and 'ThirdParty' • New semantics of 'useruid', 'username' and 'password' fields in the first ATTACH request for third-party clients only • New fields in the ATTACH response: 'useruid', 'username' • Introduced guidelines for third-parties about the generation of unique values for the 'source' field • Updated table in Managing of different protocol versions
2.4	2021-09-09	MN	36 34 12	• Deprecated `requireencryption` in SESSION. It is possible to insert this field for backward compatibility, as defined in previous versions of the protocol. From this version it will be ignored
2.5	2021-10-14	AG	15 50	• Deprecated `lang` parameter in ATTACH and MODIFYSCENE. It is possible to insert this field for backward compatibility, as defined in previous versions of the protocol. From this version it will be ignored • Fixed some minor typos in some examples
2.7	2023-11-03	MN	34 15 10 24 30	• Added new error code in the command DoAction: ERR INVALID PWD. • The optionality of the Attach response `permissions` field is made explicit • Fixed `deviceuid` field definition and related examples • Added more details about `password` field during thirdparty client association • Added new (optional) field `withvalues` as parameters of SFDISCOVERY and REGISTER commands • Enhanced `params` description in SFDISCOVERY, DOACTION e REGISTER

dat.nguyendinh@vtd.com.vn

Contents

dat.nguyendinh@vtd.com.vn

1 Introduction

This document contains the specifications of the "IP Connector" communication protocol between the following parts:

- Automation Gateway (AG)
- Automation Gateway+ (AG+)
- Third-party devices

2 Overview

2.1 Premise

The IP Connector protocol is aimed at bi-directional communication between a "server" and one or more "clients" through a (local) network connection to perform the following functions:

- Automatically identify the presence of servers on the network
- Register the client as an authenticated contact partner on the server
- Receive information on the structure and resources exported from the server
- Send commands and status requests synchronously
- Receive status change information asynchronously

The protocol also provides mechanisms to protect the communication and to manage particular situations (change of network configuration, interruption and recovery of communication, timeout etc ...) that require a correct recovery of the communication channel.

In order to be authenticated on the server, a third-party client needs the third-party unique identifier assigned by Vimar, the Third-Party Tag, and the private RSA-key agreed between Vimar and the third-party.

2.2 The communication channel

Communication takes place via a WebSocket [1] channel, in local connection, on a port established between server and client that must be protected by SSL encryption; to ensure the security of communication against possible replicas of the package. The client must always validate the session by using the provided VimarCA. In addition to encryption, the exchange of information also includes a "token" generated dynamically by the server, unique for each session and client, and renewed periodically.
The server can communicate with several clients simultaneously, each on a dedicated port (figure 1):

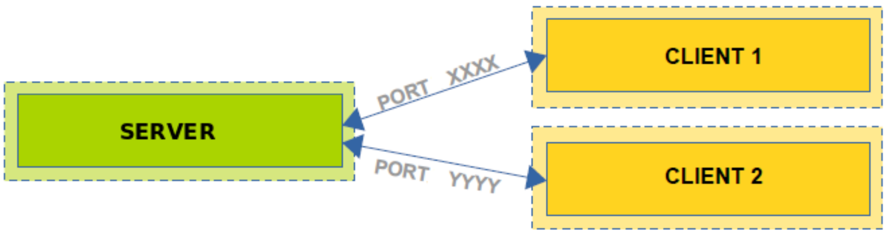

Figure 1: client-server communication schema

In addition, each client can command and "register" to receive status change notifications for sets of different objects, all consistent with the access permissions given by the server administrator to the client.

dat.nguyendinh@vtd.com.vn

In particular, the client opens a WebSocket communication with the server on the established port (see details below) at the beginning of the session; the channel created is permanent until one of the two endpoints send the close command or due to problems with reachability.

The WebSocket type channel is by its nature bidirectional.

2.3 The message format

The communication protocol (excluding the SEARCH message) provides for the encapsulation of the information in data packets in JSON format (www.json.org), a format that provides a specific syntax for data structures composed of objects, arrays and single elements, as well as being natively supported by numerous programming environments.

For a description of the message structure, see the chapter 3.1.

The packets must be UTF-8 encoded [2].

2.4 The structure of application information

Information on the subsystems exported from gateway devices is structured according to the specification of the System Functions (hereinafter SF) Vimar [3].
In the following paragraphs we will also refer to the individual System Functions Elements (hereafter SFE); for details on SFE see the SF specification.

2.5 The functions provided

The protocol foresee the use of functions characterized by requests (request) and responses (response); for each request a response must be provided from the other endpoint, with the exception of CHANGESTATUS and EXPIRE where the request can specify that it does not request the response via the `requiredresp` parameter.

The communication includes the following phases:

PHASE	DIRECTION	DESCRIPTION
SEARCH	client → server	Search the servers on the network
SESSION	client → server	Negotiate the session between client and server
ATTACH	client → server	Dual function: • Pairing between client and server, automatic creation of credentials and data necessary for access (first connection) • Reconnection
AMBIENTDISCOVERY	client → server	Request of the list of environments in which the system is structured
SFDISCOVERY	client → server	Request of the SF and SFE list that the client can manage through the communication channel

dat.nguyendinh@vtd.com.vn

REGISTER	client → server	Registration of the SFE list that the client intends to manage through the communication channel
UNREGISTER	client → server	Removal from your registration of one or more SFE that the client no longer wishes to manage
DOACTION	client → server	Sending commands to the server to perform actions on SFE
GETSTATUS	client → server	Synchronous status request on one or more SFE
CHANGESTATUS	server → client	Notification of status change to the client of one or more SFE previously registered
EXPIRE	server → client	Dual function: • Informs the client of the imminent disconnection from the server, with the consequent need to restart a new session
DETACH	client → server	Dual function: • Closes the communication session with the server • It requires the removal of the logged in user and closes the communication session with the server
KEEPALIVE	client → server	Keep the session active with the server
CREATESCENE	client → server	Create a new scenario
MODIFYSCENE	client → server	Modify a previously created scenario
DELETESCENE	client → server	Delete a previously created scenario
RETRIEVELANGDICTIONARY	client → server	Retrieve text dictionary
GETLOG	client → server	Requires the events LOG

2.6 Managing of different protocol versions

Client and Server may use different protocol versions. The following table shows how they must behave in all the cases:

CLIENT VERSION	SERVER VERSION	BEHAVIOR
`>=2.2`	`>=2.2`	The server does support third-party clients

dat.nguyendinh@vtd.com.vn

3 The communication protocol

3.1 Message structure

Each command of the IP Connector protocol (with the exception of the SEARCH command, see dedicated paragraph) has a common structure of the Json message.
A request message has the following format:

Common syntax of the request

```
{
  "type":"request",
  "function":"attach",
  "source":"8YVGVZdtjKpLq6DbnG",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"0",
  "args":[],
  "params":[]
}
```

Request parameters details:

PARAMETERS	DESCRIPTION	EXAMPLE
`type`	Message type	“request”
`function`	Required function	“attach”
`source`	Unique identifier of the sender (note 1 of 3.1.1)	“8YVGVZdtjKpLq6DbnG”
`target`	Unique identifier of the recipient (note 1 of 3.1.1)	“e6fd3d010e2341d4”
`token`	Unique session identifier, is required only for client requests to the server. Its length is 12 UTF-8 characters	“WjJkRkZGRkY=”
`msgid`	Unique identifier of the message within a communication session; to be reported in the answer	“0”
`args`	Arrays of specific objects for each command.	[]
`params`	Array of additional and specific parameters for each command.	[]

All the commands not depending on request `params` value do not perform any check on the `params` attribute and ignore its content.

A response message has the following format:

Common response syntax

```
{
  "type":"response",
  "function":"attach",
```

dat.nguyendinh@vtd.com.vn

```
   "source":"e6fd3d010e2341d4",
   "target":"8YVGVZdtjKpLq6DbnG",
   "msgid":"0",
   "error":0,
   "result":[]
 }
```

Response parameters details:

PARAMETERS	DESCRIPTION	EXAMPLE
`type`	Message type	“response”
`function`	Required function	“attach”
`source`	Unique identifier of the sender (note 1 of 3.1.1)	“e6fd3d010e2341d4”
`target`	Unique identifier of the recipient (note 1 of 3.1.1)	“8YVGVZdtjKpLq6DbnG”
`token`	Unique session identifier, only needed for client responses to the server	-
`msgid`	Unique identifier of the message; must match the one reported in the request that generated the response	“0”
`error`	Error code; 0 if there are no errors	0
`result`	Arrays of specific objects for each command.	[]

Here follow the generic return codes of the `error` parameter. The error codes specific to the various commands are available in each of the sections related to protocol commands.

ERROR CODE	MEANING
`ERR UNKNOWN FUNCTION`	Unrecognized function
`ERR INVALID SOURCE`	Value of the `source` field not consistent with the `source` field that identifies the session
`ERR INVALID TARGET`	Value of the `target` field not consistent with the DUID
`ERR INVALID TOKEN`	Value of the `token` field not consistent with the session token (in the case of `function` other than ATTACH)
`ERR INVALID TYPE`	Value of the `type` filed not allowed (other than `request` and `response`)

dat.nguyendinh@vtd.com.vn

`ERR MALFORMED MESSAGE`	The message is not in JSON format or does not respect the format of the messages and in particular does not contain one of the following fields: • `type` (string) • `function` (string) • `source` (string) • `target` (string) • `msgid` (string) • `token` (in case of messages with `type request`, string type) • `args` (in case of messages with `type request`, array type) • `params` (in case of messages with `type request`, array type) • `result` (in case of messages with `type response`, array type) • `error` (in case of messages with `type response`, integer type)

Messages that are not formatted correctly (not Json or without some keys) are ignored by the server. In this case the server will reply with a non IpConnector-formatted message containing only the error code of ERR MALFORMED MESSAGE error.

3.1.1 Notes

The unique identifier is:

- for Vimar clients
  - a unique ID device-specific (`deviceuid`) in the case of devices (AGx, MTSx, SAIG, etc ...)
- for third-party clients
  - a unique ID device-specific. It is suggested the use of Globally Unique Identifiers [4] or MAC addresses

dat.nguyendinh@vtd.com.vn

3.2 SEARCH

The first phase involves the search and identification in the local network (LAN) of one or more servers available through the use of the mDNS protocol (Multicast DNS, [5], [6]).
In particular, the client that wants to identify the available servers on the network must perform a "One-Shot Multicast DNS Query" (possibly repeated) requiring a Unicast Response (set the unicast-response bit).
The servers available on the same LAN will reply by exporting the two services related to the interaction through the IpConnector protocol:

- `_vimar-devctrl._tcp`: channel used for control by devices, mobile applications, other.

Here is an example of an answer to an mDNS query:

```
hostname = [vrag09.local]
address = [172.20.32.101]
port = [443]
txt = ["deviceuid=A32101FBB00001" "model=AG+" "mac=FF:EE:DD:CC:BB:AA"
"softwareversion=x.y.z" "protocolversion=u.v.t" "communicationmode=4"
"plantuid=ccbbaa1234567890" "plantname=Casa al mare"
"devicename=Gateway della casa al mare" "ntpmode=Client"
"uuid1=<INSTALLER UUID>" "uuid2=<ADMIN UUID>"]
```

PARAMETER	DESCRIPTION	EXAMPLE
`hostname`	Device name	vrag09.local
`address`	Server IP address	172.20.32.101
`port`	Port on which the service is available	443
`txt`	Text field containing more information:	
-	`deviceuid`: device serial number	A32101FBB00001 Composed of 14 alphanumeric digits [0..9][A..F]
-	`model`: device type • AG+ • AG	AG+
-	`mac`: device MAC address	FF:EE:DD:CC:BB:AA
-	`softwareversion`: SW version installed on the device in x.y.z format	1.0.0
-	`protocolversion`: protocol version implemented on the device in u.v.t format	1.0.0
-	`communicationmode`: communication mode supported (see notes below)	4
-	`plantuid`: unique plant identifier	ccbbaa1234567890 Composed of at most 32 alphanumeric digits [0..9][a..f]

dat.nguyendinh@vtd.com.vn

-	`plantname`: plant identification name	Casa al mare
-	`devicename`: descriptive name of the gateway, which can be set by the installer.	Gateway della casa al mare
-	`ntpmode`: operating mode of the NTP service daemon, which can be set by the installer. • Client • Server	Client
-	`uuid1`: UUID related to the installer	e5313ea20d3a
-	`uuid2`: UUID related to the administrator	e47115b32h87

Notes:

- The order of the key-value pairs shown in the example for the "txt" field is purely indicative
- The possible modes of communication are:
  1. “on demand” TCP communication channel
  2. TCP communication channel always active
  3. TCP channel with dual mode (1 and 2)
  4. WebSocket channel

The client must then exploit the information obtained from the mDNS response to perform the following steps. In particular, the client must exploit the address - port pair to execute the SESSION request to the server.

3.2.1 Error conditions

- No response within the timeout for the search cycle. In this case it must be established how many cycles (new search attempts) must be performed.

3.2.2 Notes

1. The client must be ready to receive mDNS responses from multiple servers and collect these responses and then present the result to the upper layer (possibly based on filters related to the current application procedure).

   The collection of responses can be interrupted if the client is acting within a particular application process: for example, the client is searching only for a specific device (`deviceuid`). In this case, received a response from the specific server, the search can be interrupted and the result presented to the upper layer.

2. For the search, set a timeout of N seconds.

3. The client can collect at most the responses of M devices.

4. Only one mDNS query must be performed within the same search cycle.

dat.nguyendinh@vtd.com.vn

3.3 SESSION

Once the device information has been obtained from the mDNS response, the client must open a WebSocket connection on the port indicated in the SEARCH response.

Syntax of the SESSION request

```
{
  "type":"request",
  "function":"session",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"A68hga4",
  "msgid":"0",
  "args":[
    {
      "communication":
      {
        "communicationmode":4,
        "ipaddress":"192.168.0.100",
        "ipport":0
      }
    }
  ],
  "params":[]
}
```

The `args` parameter contains a single object with the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`communication`	Object that collects communication information	see related table

The `communication` parameter is an object containing the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`communicationmode`	Communication mode that the client wants to use: 1. “on demand” TCP communication channel 2. TCP communication channel always active 3. TCP channel with dual mode (1 and 2) 4. WebSocket channel (the Websocket channel is the only currently supported mode)	1
`ipaddress`	Client LAN IP address (required but not used)	“192.168.0.100”

dat.nguyendinh@vtd.com.vn

`ipport`	IP port on which the client will accept requests from the server in the case of communication mode 1,2 or 3. In the case of a WebSocket type channel, the value must be left blank (Since currently is only available the WebSocket communication mode, the field is required but not used)	1234

Parameter `params` does not contain any object.

Syntax for SESSION response

```
{
  "type":"response",
  "function":"session",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"0",
  "error":0,
  "result":[
    {
      "communication":
      {
        "ipport":1234
      }
    }
  ]
}
```

The `result` parameter contains a single object composed by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`communication`	Object that collects communication information	see related table

The `communication` parameter is an object containing the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`ipport`	Port available on server where the client can proceed with ATTACH	1234

Here follow the possible return codes specific to the `error` parameter:

ERROR CODE	MEANING

dat.nguyendinh@vtd.com.vn

`ERR MALFORMED ARGS`	One of the following fields is not present within the first object of the `args` array: • `communication` • `communicationmode` (within `communication` with integer type) • `ipport` (within `communication` with integer type) • `ipaddress` (within `communication` with string type)
`ERR SESSION`	Unable to retrieve a valid port on which to perform ATTACH (maximum number of sessions reached or DeviceUpdate in progress)
`ERR INVALID COMM MODE`	The selected `communicationmode` is not supported (it is different from WebSocket)

3.3.1 Error conditions

- It may not be possible to get a session from the server to reach the maximum number of simultaneously connected clients. No more than 20 active connections are allowed at the same time.
- The client may request a communication mode that is not available / not supported by the server.

3.3.2 Notes

- Each client dynamically gets a different port assigned by the server and generally different for each session started by the same client.
- A session could possibly use the same port on which the server is also listening (port of the SESSION command)
- As specified in the paragraph 2.2, the only communication mode implemented is option 4.

dat.nguyendinh@vtd.com.vn

3.4 ATTACH

Once the SEARCH and SESSION are done to receive the IP address of the server and the port dedicated to the specific client, you can proceed with the association sequence between client and server or login, in case the first association has already been made.

The first association of a client on the server must be previously enabled through the authorization of the user administrator or installer; for third-party clients, only the installer can enable the client association and must provide the client with an ad-hoc setup code in order to complete the association.

The syntax of the ATTACH request is the same for both the first association and the subsequent ones.

Syntax of the ATTACH request

```
{
  "type":"request",
  "function":"attach",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"dFvC395h6UsR",
  "msgid":"0",
  "args":[
    {
      "credential":
      {
        "username":"aThirdPartyTag",
        "useruid":"C7JPKC96G5MQ5844",
        "password":"eXuiYTFzf245352Y"
      },
      "clientinfo":
      {
        "manufacturertag":"aThirdPartyTag",
        "clienttag": "thirdpartyapp",
        "sfmodelversion":"1.0.0",
        "protocolversion":"2.5"
      },
      "communication":
      {
        "ipaddress":"192.168.0.100"
      }
    }
  ],
  "params":[]
}
```

The `args` parameter contains one only object with the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`credential`	Object that collects user credentials	see related table

dat.nguyendinh@vtd.com.vn

`clientinfo`	Object that collects client information	see related table
`communication`	Object that collects communication information	see related table

The `credential` parameter is an object containing the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`username`	The field must be filled with the unique identifier dedicated to the third-party (Third-Party Tag).	`"aThirdPartyTag"`
`useruid`	Unique identifier used by the user to access the gateway. For third-party clients the identifier is assigned by the server in the first association (details in Note for third-party clients). It is a string of at most 36 alphanumeric characters	"C7JPKC96G5MQ5844"
`password`	Access password associated with the user. During the first association the field must be filled with the encrypted setup code provided by installer. For third-party clients, the value must be encrypted even in subsequent associations (details in Note for third-party clients).	"ncpaiusfn87324ng"

3.4.1 Note for third-party clients

In the association phase, i.e. in the first ATTACH:

- the `token` field must be filled with random strings
- the `username` field must be filled with the identifier assigned to the third-party (Third-Party Tag);
- the `password` field must be filled with the setup code provided by installer; this code must be signed with the 2048 bit private RSA-key of the third-party, using PKCS#1 v1.5. Then the result must be base-64 encoded.
- the `useruid` field must be empty.

In the ATTACH response, the server will send the `useruid` and `password` generated for the client, together with the `username` preliminary defined by the installer and a `token` valid for the whole session. On the server the new association is immediately confirmed.
From the next ATTACH requests and outside the first binding process, the client will have to insert the value obtained by the server in the first binding for the `useruid` and `password` fields (the `password` field must always be encrypted and encoded to base-64). The `token` field must always be filled with a random string•.

In case of ATTACH request to session already active, a response with a suitable error code is sent.

The `clientinfo` parameter is an object containing the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`manufacturertag`	Manufacturer's identification string. Optionally, Third-Party Tag can be used here	"aThirdPartyTag"

dat.nguyendinh@vtd.com.vn

`clienttag`	Application name: • thirdpartyapp	"thirdpartyapp"
`sfmodelversion`	Version of the client's SF model, in X.Y.Z format (required but not used)	"1.0.0"
`protocolversion`	Client IpConnector protocol version	"2.2"

The `communication` parameter is an object containing the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`ipaddress`	Client LAN IP address (required but not used)	"192.168.0.100"

The `params` parameter contains no objects.
In response to the ATTACH request, once the credentials have been validated the new client is enabled.

At this point a response is given to the client, in the following format:

Syntax of the response to the ATTACH request

```
{
  "type":"response",
  "function":"attach",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"0",
  "error":0,
  "result":[
    {
      "serverinfo":
      {
        "manufacturertag":"aThirdPartyTag",
        "model":"AG+",
        "deviceuid": "A32101FBB00001",
        "mac":"FF:EE:DD:CC:BB:AA",
        "sfmodelversion":"1.0.0",
        "softwareversion":"1.0.0",
        "firmwareversion":"1.0.0",
        "dbmodelversion":"1.0.0",
        "hardwareversion":"1.0.0",
        "hash":"13FE",
        "dictionaryHash":"DICTIONARY_HASH",
        "protocolversion":"2.2"
      },
      "secureinfo":
      {
        "publickey":"-----BEGIN PUBLIC KEY-----MIIBIjANBgkqhkiG9w...."
      },
      "plantname":"Casa al lago",
```

dat.nguyendinh@vtd.com.vn

```
        "plantuid":"ccbbaa1234567890",
        "usertype":"Administrator",
        "password":"nsoeLApw52043ch7",
        "useruid":"c6b963a3-f935-409a-8725-c79cd5e972a8",
        "username":"HUB Soggiorno",
        "token":"WjJkRkZGRkY=",
        "sfcategory":["Plant","BigData","Scene","LogicProgram",
                      "ConfGateway","InfoGateway"],
        "permissions":["SCENE_PERSONALIZE","CLIMATE_CONTROL"]
      }
    ]
  }
```

The `result` parameter contains a single object composed of parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`serverinfo`	Object that collects generic server information	see related table
`secureinfo`	Object that collects server security information	see related table
`plantname`	Plant name	“Casa al lago”
`plantuid`	Plant unique-id	“ccbbaa1234567890”
`usertype`	User type • ThirdParty	"ThirdParty"
`password`	Password generated by the server in first association, then with random value	“eXuiYTFzf245352Y”
`useruid`	Unique identifier used by client to access the gateway. For third-party clients it is generated by server in the first association; in subsequent associations its value is equal to the one sent by the client	"c6b963a3-f935-409a-8725-c79cd5e972a8=”
`username`	Client (device) name. For third-party clients, it is assigned by installer	"HUB Soggiorno”
`token`	Session token to send in all messages to the server	“WjJkRkZGRkY=”

dat.nguyendinh@vtd.com.vn

`sfcategory`	Array of SF categories that the user can access: • Plant: exports the SFs relating to the home automation system • BigData: exports the SF for the analysis of data obtained from the home automation system, such as information on daily, weekly, monthly energy consumption, ... • LogicProgram: exports the SFs for the activation of the logic programs and the modification of any associated timers • ConfGateway: exports the SFs for the modification of some configuration parameters of the gateway, including for example the list of associated users • InfoGateway: exports the SFs to read the gateway's board data	[Plant, Scene, Info-Gateway]
`permissions` (opzional field)	Array of User permissions: • `SCENE_PERSONALIZE` • `PROGRAM_PERSONALIZE` • `CLMATE_PERSONALIZE` • `ENERGY_PERSONALIZE` • `LIGHT_CONTROL` • `SHUTTER_CONTROL` • `CLIMATE_CONTROL` • `LOAD_CONTROL` • `IRRIGATION_CONTROL` • `PROGRAM_CONTROL` • `ACCESS_CONTROL` • `ACTUATOR_CONTROL` • `CONSUMPTION_CONTROL` • `MULTIMEDIA_CONTROL` • `SENSOR_CONTROL` • `SCENE_CONTROL` • `LOGIN` • `DATETIME_CONTROL` • `UNDEFINED` • `LIGHT_PERSONALIZE` • `IRRIGATION_PERSONALIZE`	`[SCENE_PERSONALIZE, CLIMATE_CONTROL]`

The `serverinfo` parameter is an object composed by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`manufacturertag`	Manufacturer's identification string	“aThirdPartyTag”

dat.nguyendinh@vtd.com.vn

`model`	Server name. For further details see the description of the same field in the SEARCH function	"AG+"
`deviceuid`	Unique identifier that identify the device and is equal to its serial number, as reported in the SEARCH	"A32101FBB00001" Composed of 14 alphanumeric digits [0..9][A..F]
`mac`	Device MAC address	"FF:EE:DD:CC:BB:AA"
`sfmodelversion`	Version of the SF model of the server, in the format X.Y.Z	"1.0.0"
`softwareversion`	Server software versione, in X.Y.Z format	"1.0.0"
`firmwareversion`	Server firmware version, in X.Y.Z format	"1.0.0"
`dbmodelversion`	Model version of the server database, in X.Y.Z format	"1.0.0"
`hardwareversion`	Server hardware version, in X.Y.Z format	"1.0.0"
`hash`	Plant Hash, a variation of it indicates a modification of the plant	"13FE". In case of SAIGx, the SAI category SFs are not considered in the calculation
`dictionaryHash`	Dictionary Hash can be useful on client side to choose if call or not the RETRIEVELANGDICTIONARY API	"13FE"

The `secureinfo` parameter is an object composed by the following parameters:

PARAMETER	**DESCRIPTION**	**EXAMPLE**
`publickey`	Reserved field. It serves some later purpose	"——BEGIN PUBLIC KEY—— MIIBIjANBgkqhkiG9... ——END PUBLIC KEY——"

The following are the possible return codes specific to the `error` parameter:

ERROR CODE	**MEANING**

dat.nguyendinh@vtd.com.vn

`ERR MALFORMED ARGS`	Within the first object of the `args` array one of the following fields is not present: • `credential` • `clientinfo` • `communication` • `username` (within `credential` with string type) • `useruid` (within `credential` with string type) • `password` (within `password` with string type) • `manufacturertag` (within `clientinfo` with string type) • `clienttag` (within `clientinfo` with string type) • `sfmodelversion` (within `clientinfo` with string type) • `ipaddress` (within `communication` with string type) • `protocolversion` (within `clientinfo` in the request and `serverinfo` in the response. String type. Ex. 1.25)
`ERR SESSION ALREADY STARTED`	In the current session the login has already been performed
`ERR PERMISSION DENIED`	The login returned a denied permission error
`ERR INVALID PWD`	The password provided is not valid for login
`ERR INVALID CLIENT TAG`	The login produced an invalid `clienttag` error
`ERR USER NOT FOUND`	The login returned a user not found error
`ERR SYSTEM BLOCK`	Generic error returned in all cases other than the previous ones
`ERR SYSTEM LOADING`	System is still booting up and is not ready

3.4.2 Notes

The client must check the compatibility with the server in terms of SW and protocol version (details in Managing of different protocol versions).

dat.nguyendinh@vtd.com.vn

3.5 AMBIENTDISCOVERY

This operation allows the client to request a complete list of the environments available in the system.

Syntax of the AMBIENTDISCOVERY request

```
{
  "type":"request",
  "function":"ambientdiscovery",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"1",
  "args":[],
  "params":[]
}
```

The `args` parameter contains no objects.
The `params` parameter contains no objects.

AMBIENTDISCOVERY response syntax

```
{
  "type":"response",
  "function":"ambientdiscovery",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"1",
  "error":0,
  "result":[
    {
      "idambient":0,
      "name":"Primo piano",
      "dictKey":"9999",
      "hash":"ADBF1",
      "idparent":0
    },
    {
      "idambient":1,
      "name":"Soggiorno",
      "dictKey":"8888",
      "hash":"BBE1",
      "idparent":0
    },
    {
      "idambient":2,
      "name":"Cucina",
      "dictKey":"7777",
      "hash":"234A",
      "idparent":0
    },
```

dat.nguyendinh@vtd.com.vn

```
    {
      "idambient":3,
      "name":"Angolo giochi",
      "dictKey":"6666",
      "hash":"BB87",
      "idparent":1
    }
  ]
}
```

The `result` parameter contains a list of objects described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idambient`	It uniquely identifies the environment. The root environment always has an id of 0.	0
`name`	Name of the environment, chosen during configuration	"First floor"
`dictKey`	Id in the dictionary for the environment name	"9999"
`hash`	Identify the environment version. It is updated following the changes of the SFs contained in the environment	"ADBF1"
`idparent`	Unique identifier of the parent environment. The root environment always has itself as a parent environment.	0

If no environment has been defined, the response to the AMBIENTDISCOVERY function will return the parameter texttt result without objects.

The following are the possible return codes specific to the `error` parameter:

ERROR CODE	MEANING
`ERR UNKNOWN FUNCTION`	Function not recognized for this type of device
`ERR READING DB`	Error in reading the plant locations
`ERR PLANT NOT CONFIGURED`	No locations found inside plant

dat.nguyendinh@vtd.com.vn

3.6 SFDISCOVERY

The SF function allows the client to request the list of SF in a specific SF category. Only for the categories `Plant` and `LogicProgram` must be specified the environment or the list of environments of interest.

Syntax of SFDISCOVERY request

```
{
  "type":"request",
  "function":"sfdiscovery",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"2",
  "args":[
    {
      "sfcategory":"Plant"
    }
  ],
  "params":[
    {
      "idambient":[2,1,3]
    }
  ]
}
```

The `args` parameter contains one only object described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`sfcategory`	Identification string of one of the SF categories exposed by the server to the client through the response to the ATTACH function. See this function for the definition of the categories	Plant

The `params` parameter can be empty or contain a single object with the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idambient`	Needed for `sfcategory` equal to `Plant` or `LogicProgram`. Array of identifiers of environments whose SFs are to be known. Leaving the array empty, a discovery of all the SFs of the plant will be performed, related to the `Plant` or `LogicProgram` category	[2,1,3]

dat.nguyendinh@vtd.com.vn

`withvalues` (opzional field)	Boolean parameter to request or not the `value` and `enable` fields of SFEs in the response A request with `withvalues : false` generates a response in which the items of the elements array for each SF are missing the `value` and `enable` fields If the parameter is missing in the request, it is assumed to be `true`.	false

In the case of system discovery, the related response will take the form described below.

Syntax of the SFDISCOVERY response

```
{
  "type":"response",
  "function":"sfdiscovery",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"2",
  "error":0,
  "result":[
    {
      "idambient":2,
      "sf":[
        {
          "idsf":122,
          "sftype":"SF_Light",
          "sstype":"SS_Light_Switch",
          "name":"Lampdario centrale",
          "dictKey":"1111",
          "elements":[
            {
              "sfetype":"SFE_Cmd_OnOff",
              "value":"",
              "enable":true
            },
            {
              "sfetype":"SFE_State_OnOff",
              "value":"Off",
              "enable":true
            }
          ]
        }
      ]
    },
    {
      "idambient":1,
      "sf":[
        {
          "idsf":134,
```

dat.nguyendinh@vtd.com.vn

```
      "sftype":"SF_Light",
      "sstype":"SS_Light_Switch",
      "name":"Luce Soggiorno",
      "dictKey":"2222",
      "elements":[
        {
          "sfetype":"SFE_Cmd_OnOff",
          "value":"",
          "enable":true
        },
        {
          "sfetype":"SFE_State_OnOff",
          "value":"On",
          "enable":true
        }
      ]
    },
    {
      "idsf":135,
      "sftype":"SF_Light",
      "sstype":"SS_Light_Dimmer",
      "name":"lampada TV",
      "dictKey":"3333",
      "elements":[
        {
          "sfetype":"SFE_Cmd_OnOff",
          "value":"",
          "enable":true
        },
        {
          "sfetype":"SFE_State_OnOff",
          "value":"On",
          "enable":true
        },
        {
          "sfetype":"SFE_Cmd_Brightness",
          "value":"",
          "enable":true
        },
        {
          "sfetype":"SFE_State_Brightness",
          "value":"50",
          "enable":true
        },
        {
          "sfetype":"SFE_State_FailureAlarm",
          "value":"No alarm",
          "enable":true
        }
```

dat.nguyendinh@vtd.com.vn

```
          ]
        },
        {
          "idsf":155,
          "sftype":"SF_Shutter",
          "sstype":"SF_Shutter_WithoutPosition",
          "name":"Tapparelle Soggiorno",
          "dictKey":"4444",
          "elements":[
            {
              "sfetype":"SFE_Cmd_ShutterWithoutPosition",
              "value":"",
              "enable":true
            },
            {
              "sfetype":"SFE_State_ShutterWithoutPosition",
              "value":"Idle",
              "enable":true
            }
          ]
        }
      ]
    },
    {
      "idambient":3,
      "sf":[]
    }
  ]
}
```

The `result` parameter consists of a list of objects defined by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idambient`	Unique identifier of the environment	2
`sf`	Array of SF objects	see related table

The `sf` parameter consists of a list of objects defined by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	Unique identifier of the SF	122
`sftype`	Type of SF, for a complete list see the SF specification	"SF_Light"
`sstype`	Realization of SF. For a complete list see the SF specification	"SS_Light_Switch"
`name`	Name of the SF	"Central chandelier"
`dictKey`	Id in the dictionary for the name of the SF	"3333"
`elements`	Array of SFE objects	see relative table

The `elements` parameter consists of a list of objects defined by the following parameters:

dat.nguyendinh@vtd.com.vn

PARAMETER	DESCRIPTION	EXAMPLE
`sfetype`	Type of SFE. For a complete list, see the SF specification	"SFE_Cmd_OnOff"
`value`	Value of SFE. Command SFE do not have a value	""
`enable`	Value indicating whether the SFE is enabled or not	true

The return codes are shown in the appendix.

On the other hand, if the discovery concerns any other category of SF, the relative response can be described as in the following example, which takes into account the category `ConfGateway`:

Syntax of the SFDISCOVERY response

```
{
  "error": 0,
  "function": "sfdiscovery",
  "msgid": "2",
  "result": [
    {
      "elements": [
        {
          "enable": true,
          "sfetype": "SFE_State_NTPServerClient",
          "value": "Server"
        },
        {
          "enable": true,
          "sfetype": "SFE_State_Timezone",
          "value": "Europe/Skopje"
        }
      ],
      "idsf": 50000,
      "name": "SF_DateTimeConfig",
      "sftype": "SF_DateTimeConfig",
      "sstype": "SS_DateTimeConfig"
    }
  ],
  "source": "e6fd3d010e2341d4",
  "target": "8YVGVZdtjKpLq6DbnG7WQ9km",
  "type": "response"
}
```

The `result` parameter consists of a list of objects defined by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	Unique identifier of the SF	122

dat.nguyendinh@vtd.com.vn

`sftype`	Type of SF, for a complete list see the SF specification	"SF_DateTimeConfig"
`sstype`	Realization of SF, for a complete list see the SF specification	"SS_DateTimeConfig"
`name`	Name of the SF	"SF_DateTimeConfig"
`elements`	Array of SFE objects	see related table

The `elements` parameter consists of a list of objects defined by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`sfetype`	Type of SFE. For a complete list, see the SF specification	"SFE_State_Timezone"
`value`	Value of SFE. SFE type command does not have a value	""
`enable`	Value indicating whether the SFE is enabled or not	true

Here follows the possible return codes relted to the `error` parameter:

ERROR CODE	MEANING
`ERR PERMISSION DENIED`	Permission denied on the category of System Function required
`ERR DATA`	Unable to retrieve one of the requested environments according to the `idambient` provided (for SFDiscovery on category `Plant`)
`ERR MALFORMED ARGS`	The error code can be returned if one of the following fields is not present in the first object of the `args` array: • `sfcategory` (string)
`ERR MALFORMED PARAMS`	In case of SFDiscovery on `Plant` or `LogicProgram` category the error code is returned when one of the following fields is not present in the first object inside the `params` array: • `idambient` (array of integers)

dat.nguyendinh@vtd.com.vn

3.7 REGISTER

Once received SFs and SFEs by SFDISCOVERY, the client can subscribe through REGISTER function to one or more SFEs in order to receive asynchronous change notifications (CHANGESTATUS) of the SFE state ((`value` ed `enable` parameters).

The registration is valid only within the current session; in any new session the client needs to register again for the necessary SFEs.

Syntax of the REGISTER request

```
{
  "type":"request",
  "function":"register",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"3",
  "args":[
    {
      "idsf":155,
      "sfetype":[]
    },
    {
      "idsf":135,
      "sfetype":["SFE_State_OnOff","SFE_Cmd_Brightness","SFE_State_Brightness"]
    }
  ],
  "params":[]
}
```

The `args` parameter contains a list of objects described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	Unique identifier of the System Function to register	135
`sfetype`	SFE arrays, related to the SF with identifier `idsf`, to register. The empty array indicates the registration request to all the SFEs contained in the SF	["SFE_State_OnOff", "SFE_Cmd_Brightness", "SFE_State_Brightness"]

The `params` parameter can be empty or contain a single object with the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE

dat.nguyendinh@vtd.com.vn

`withvalues` (opzional field)	Boolean parameter to request or not the `value` and `enable` fields of the SFEs required for subscription A request with `withvalues : false` generates a response with an empty result array A request with `withvalues : true` generates a response in which the result array contains all the SFEs with ’`get`’ access mode and in the same format as the result field in a GetStatus response for the same SFEs. If the parameter is missing in the request, it is assumed to be `false`.	true

Syntax of the REGISTER response

```
{
  "type":"response",
  "function":"register",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"3",
  "error":0,
  "result":[]
}
```

The `result` parameter contains no objects.

The following are the possible return codes specific to the `error` parameter:

ERROR CODE	**MEANING**
ERR MALFORMED ARGS	Within one of the objects within the `args` array, one of the following fields is not present: • `idsf` (integer) • `sfetype` (array) or the array `args` is empty.
ERR DATA	One of the required System Function Elements is not available or it has failed to recover all the System Function Elements within the System Function having id `idsf` (in the case of empty array `sfetype`)

dat.nguyendinh@vtd.com.vn

3.8 UNREGISTER

Like REGISTER, the UNREGISTER function allows the client to cancel its subscription to one or more SFEs previously registered. In this way, the client no longer receives notifications of state changes (CHANGESTATUS) of those SFEs.

Syntax of the UNREGISTER request

```
{
  "type":"request",
  "function":"unregister",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"8",
  "args":[
    {
      "idsf":155,
      "sfetype":[]
    },
    {
      "idsf":135,
      "sfetype":["SFE_State_OnOff","SFE_Cmd_Brightness","SFE_State_Brightness"]
    }
  ],
  "params":[]
}
```

The `args` parameter contains a list of objects described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	Unique identifier of the System Function to unregister	135
`sfetype`	SFE array to unregister. The empty array indicates the request to unregister all the SFEs contained in the SF with identifier `idsf`	["SFE_State_OnOff", "SFE_Cmd_Brightness", "SFE_State_Brightness"]

The `params` parameter contains no objects.

Syntax of the UNREGISTER response

```
{
  "type":"response",
  "function":"unregister",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"8",
  "error":0,
  "result":[]
```

dat.nguyendinh@vtd.com.vn

```
}
```

The `result` parameter contains no objects.

The following are the possible return codes specific to the `error` parameter:

ERROR CODE	MEANING
`ERR MALFORMED ARGS`	In the objects of the `args` array, one of the following fields is not present: • `idsf` (integer) • `sfetype` (array) or the array `args` is empty
`ERR DATA`	The required System Function Element is not available or the recovery of all System Function Elements within the System Function having id `idsf` failed (in the case of empty array `sfetype`)

dat.nguyendinh@vtd.com.vn

3.9 DOACTION

The DOACTION function allows the client to perform an operation on one or more SFE with 'set' access mode (see the SF specification) of any SFs identified by SFDISCOVERY. Hence, it is possible to append multiple commands.

For multiple commands the whole request must be valid. If the request is partially malformed the whole request is discarded and none of the operations is performed.

Since the order of the objects in the `args` parameter does not necessarily define the order of execution, the client is recommended to send multiple DOACTIONs if it needs to execute the commands in a pre-established order.

Syntax of the DOACTION request

```
{
  "type":"request",
  "function":"doaction",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"5",
  "args":[
    {
      "idsf":155,
      "sfetype":"SFE_Cmd_ShutterWithoutPosition",
      "value":"Up"
    },
    {
      "idsf":135,
      "sfetype":"SFE_Cmd_OnOff",
      "value":"On"
    },
    {
      "idsf":135,
      "sfetype":"SFE_Cmd_Brightness",
      "value":"30"
    }
  ],
  "params":[]
}
```

The `args` parameter contains a list of objects described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	identification of the SF related to the element to be controlled.	
`sfetype`	SFE to be commanded.	

dat.nguyendinh@vtd.com.vn

`value`	string containing the value to be set.	`{ ... "value":"95" }, { ... "value":"On" }`

The `params` parameter contains no objects.

Syntax of the DOACTION response

```
{
  "type":"response",
  "function":"doaction",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"5",
  "error":0,
  "result":[]
}
```

In case of error (that is, a value other than `0` of the `error` parameter), none of the requested actions is performed by the server (that is, in the case of multiple actions in the same DOACTION, the failure of one implies a failure of the whole block of actions).
The `result` parameter does not contain any data.

Here follow the possible return codes related to the `error` parameter:

ERROR CODE	**MEANING**
`ERR MALFORMED ARGS`	In one fo the objects of the `args` array, one of the following fields is not present: • `idsf` (integer) • `sfetype` (string) • `value` (string) or the array `args` is empty.
`ERR DATA`	The required System Function Element is not available
`ERR SYSTEM BLOCK`	Failure of write access for a given System Function Element
`ERR ELEMENT VALUE`	The value (`value`) inserted in the System Function Element is malformed
`ERR INVALID CONTEXT`	Failure of write access due to conditions in the system that prevent modification

dat.nguyendinh@vtd.com.vn

3.10 GETSTATUS

The GETSTATUS function allows the client to request state updates (`value` ed `enable` parameters) on one or more SFEs of any SFs identified by SFDISCOVERY. State update can be requested only on SFE having 'get' access mode (see the SF specification).

Syntax of the GETSTATUS request

```
{
  "type":"request",
  "function":"getstatus",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"4",
  "args":[
    {
      "idsf":155,
      "sfetype":[]
    },
    {
      "idsf":135,
      "sfetype":["SFE_State_OnOff","SFE_State_Brightness"]
    }
  ],
  "params":[]
}
```

The `args` parameter contains a list of objects described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	id of the SF, whose elements (SFE) the status is required.	155
`sfetype`	Array of SFEs to request state. If an empty array is passed, the status of all the elements of the requested SF is returned.	["SFE_State_OnOff", "SFE_State_Brightness"]

The `params` parameter contains no objects.

Syntax of the GETSTATUS response

```
{
  "type":"response",
  "function":"getstatus",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"4",
  "error":0,
  "result":[
```

dat.nguyendinh@vtd.com.vn

```
      {
        "idsf":155,
        "elements":[
          {
            "sfetype":"SFE_Cmd_ShutterWithoutPosition",
            "value":"",
            "enable":true
          },
          {
            "sfetype":"SFE_State_ShutterWithoutPosition",
            "value":"idle",
            "enable":true
          }
        ]
      },
      {
        "idsf":135,
        "elements":[
          {
            "sfetype":"SFE_State_OnOff",
            "value":"On",
            "enable":true
          },
          {
            "sfetype":"SFE_State_Brightness",
            "value":"95",
            "enable":true
          }
        ]
      }
    ]
  }
```

dat.nguyendinh@vtd.com.vn

The information returned in the `result` parameter is an array of objects, each related to one of the required system functions. For each SF, another array of objects is returned, representing an SFE with its enabling status and value.

In particular, the parameters of each object of the `result` parameter are:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	id of the SF related to the SFE requested.	
`elements`	required SFE arrays, related to the specified SF	

The parameters of each object of the `elements` parameter are:

PARAMETER	DESCRIPTION	EXAMPLE
`idsfe`	id of each SFE required.	

`value`	string representing the current value of the SFE required. If the SFE represents a command, the value parameter loses its meaning and therefore a null string will be returned. This parameter is always present.	
`enable`	Boolean value that specifies the enabling status or not of the requested SFE.	

The following are the possible return codes specific to the `error` parameter:

ERROR CODE	**MEANING**
`ERR MALFORMED ARGS`	In one of the objects of the `args` array, one of the following fields is not present: • `idsf` (integer) • `sfetype` (string) or the array `args` is empty.
`ERR DATA`	The required System Function Element is not available

dat.nguyendinh@vtd.com.vn

3.11 CHANGESTATUS

The call is made by server to client to send notifications (CHANGESTATUS) in real-time (and in an "asynchronous" way) of state changes on each of the elements (SFE) the client has subscribed to in the current session.

Syntax of the CHANGESTATUS request

```
{
  "type":"request",
  "function":"changestatus",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"6",
  "args":[
    {
      "idsf":155,
      "elements":[
        {
          "sfetype":"SFE_State_ShutterWithoutPosition",
          "value":"moving",
          "enable":true
        }
      ]
    }
  ],
  "params":[
   {
      "requiredresp":false
   }
  ]
}
```

The information about the elements contained in `args` are passed with the same structure of the response to the GETSTATUS function.

The `params` parameter contains a single object described by:

PARAMETER	DESCRIPTION	EXAMPLE
`requiredresp`	Boolean parameter indicating whether or not the changestatus requires a response.	true

The expected syntax for the answer is as follows:

Syntax of the CHANGESTATUS response

```
{
  "type":"response",
  "function":"changestatus",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
```

dat.nguyendinh@vtd.com.vn

```
  "msgid":"6",
  "error":0,
  "result":[]
}
```

The `result` parameter does not contain any data. The return codes of the `error` parameter are shown in the appendix.

dat.nguyendinh@vtd.com.vn

3.12 EXPIRE

This call allows the server to close the communication session in progress with the client.

Syntax of EXPIRE request

```
{
  "type":"request",
  "function":"expire",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"10",
  "args":[
    {
      "reason":1,
      "value":10
    }
  ],
  "params":[
    {
      "requiredresp":false
    }
  ]
}
```

The `args` parameter contains an object described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`reason`	Reason why the disconnection is imposed by server. The number of clients receiving the same EXPIRE (target clients) depends on the reason.	see table
`value`	Generic value. Its meaning depends on the `reason`	

The `params` parameter contains an object described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`requiredresp`	Indicates whether or not a response from the client is required	true

The table presents the parameter `reason`. For each possible value, numberof target clients are reported:

- BROADCAST: all connected clients;
- MULTICAST: some connected clients;
- UNICAST: only the receiver client.

PARAMETER	DESCRIPTION	TARGET CLIENTS

dat.nguyendinh@vtd.com.vn

1	Timeout: more than 120s passed since the last message sent by the client. The client can execute a new SESSION only after `value` seconds after receiving the EXPIRE. If `value` = 0 is possible to try again immediately.	UNICAST
2	Token renewal: the client can execute a new SESSION only after `value` seconds after receiving the EXPIRE. If `value` = 0 it is possible to immediately try again.	UNICAST
3	Hash of the plant changed: the client can execute a new SESSION only after `value` seconds after receiving the EXPIRE. If `value` = 0 is possible to try again immediately.	BROADCAST, if plant changed UNICAST, if permissions or visibile environments changed
4	User disabled: The user is no longer enabled to interact with the server.	UNICAST
6	Server Restart: The client will be able to execute a new SESSION only after `value` seconds after receiving the EXPIRE.	BROADCAST
8	Hash of the plant changed (scenes): after a CREATESCENE, MODIFYSCENE or DELETESCENE command, the client can execute a new SESSION only after `value` seconds after receiving the EXPIRE. If `value` = 0 is possible to try again immediately.	BROADCAST
9	Logic Program update: the client can execute a new SESSION only after `value` seconds after receiving the EXPIRE. If `value` = 0 is possible to try again immediately.	BROADCAST

Syntax of the EXPIRE response

```
{
  "type":"response",
  "function":"expire",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"10",
  "error":0,
  "result":[]
}
```

After an EXPIRE it is necessary to carry out a new sequence of commands, dependent on `reason`.

dat.nguyendinh@vtd.com.vn

The `result` parameter does not contain any data. The return codes of the `error` parameter are shown in the appendix.

dat.nguyendinh@vtd.com.vn

3.13 DETACH

The DETACH function allows the client to close the authenticated working session. It can also ask the gateway (in addition to closing the session in progress) the removal of the user, currently connected and credited through the application, from the list of users of the gateway.

After running the DETACH function, the session token is no longer valid.

Syntax of the DETACH request

```
{
  "type":"request",
  "function":"detach",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"9",
  "args":[
    {
      "user":"logout"
    }
  ],
  "params":[]
}
```

The `args` parameter contains an object described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`user`	Reasons why the client disconnects: • `logout` - closing of the current session • `remove` - closing the current session and removing the user from the gateway user list. Once removed, the installer intervention is needed in order to run again the client-server association phase (see Note for third-party clients).	"logout"

The `params` parameter contains no objects.

Syntax of the DETACH response

```
{
  "type":"response",
  "function":"detach",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"9",
  "error":0,
  "result":[]
```

dat.nguyendinh@vtd.com.vn

```
}
```

The `result` parameter contains no objects.

After a DETACH it is required to perform a SESSION and an ATTACH, to create a new session token.

Here follows the possible return codes related to the `error` parameter:

ERROR CODE	**MEANING**
`ERR_REMOVING_USER`	Failure to remove user from user list in case of `remove` flag

dat.nguyendinh@vtd.com.vn

3.14 KEEP ALIVE

If the server does not receive any request from the client in more than 120s it sends an EXPIRE message, which closes the current session.
If the client has no specific requests, it can send a KEEPALIVE request to prevent closure and extend by 120s the current session.
Thereby the KEEPALIVE requests allow the client to keep the session active as long as needed.

Syntax of the KEEPALIVE request

```
{
  "type":"request",
  "function":"keepalive",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"12",
  "args":[],
  "params":[]
}
```

The `args` and `params` parameters do not contain objects.

KEEPALIVE response syntax

```
{
  "type":"response",
  "function":"keepalive",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"12",
  "error":0,
  "result":[]
}
```

The `result` parameter contains no objects.
The error codes of the `error` parameter are shown in the appendix.

dat.nguyendinh@vtd.com.vn

3.15 CREATESCENE

The CREATESCENE function allows the client to create a new scenario.
A scenario is a snapshot of some SFE states (parameter `value`) taken by server when the creation request is received; the SFEs are given by the set of SFs selected by client in the creation request.

At any time after creation, such states can be re-actuated all at once by 'recalling' the scenario, i.e. by sending a DOACTION request on the scenario SF. The scenario SF is generated by server on scenario creation and - as any SF - is not automatically deleted after session closure.

In the creation request, the client can also define a set of activators, that is a set of plant events able to recall the scenario, (e.g. 'button X pressed', 'door Y opened', ...).

Syntax of the CREATESCENE request

```
{
  "type":"request",
  "function":"createscene",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"13",
  "args":[
    {
      "idambient":3,
      "name":"Relax",
      "sflist":[155,135,166],
      "activatorlist":[
       {
         "idsf":235,
         "sfetype":"SFE_Cmd_DownKey_SceneActivator",
         "value":"Execute"
       }
      ]
    }
  ],
  "params":[]
}
```

The `args` parameter contains a single object with the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idambient`	Unique identifier of the environment for the created scenario	3
`name`	Scenario name	"Relax"
`sflist`	SF identifier set. The SFEs captured by the scenario are given by this set.	155,135,166

dat.nguyendinh@vtd.com.vn

`activatorlist`	Activator list. An activator is an SFE of a specific class of SFs: the SF_SceneActivator. Note that the availability of activators is not guaranteed for any plant; details on activators can be found in the SF specification.	see table

The `activatorlist` parameter consists of a list of objects (activators) defined by the following parameters:

PARAMETER	**DESCRIPTION**	**EXAMPLE**
`idsf`	Identifier of SF where `name` is SF_SceneActivator	235
`sfetype`	Type of activator SFE	"SFE_Cmd_DownKey_SceneActivator"
`value`	Constant string	"Execute"

The parameters `sflist` and `activatorlist` are optional and can be omitted.
The `params` parameter contains no objects.

dat.nguyendinh@vtd.com.vn

Syntax of the CREATESCENE response

```
{
  "type":"response",
  "function":"createscene",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"13",
  "error":0,
  "result":[
    {
      "idsf":168,
      "sftype":"SF_Scene",
      "sstype":"SS_Scene_Executor",
      "name":"Relax"
    }
  ]
}
```

The `result` parameter consists of a object defined by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	Unique identifier of the scenario SF	168
`sftype`	Type of SF (constant)	“SF_Scene”
`sstype`	Realization of SF	“SS_Scene_Scene”
`name`	Name of the scenario SF	“Relax”

Here follows the possible return codes related to the `error` parameter:

ERROR CODE	MEANING
`ERR MALFORMED ARGS`	In the first object of the `args` array, one of the following fields is not present: • `idambient` (integer) • `name` (string)
`ERR PERMISSION DENIED`	The user does not have permission `SCENE PERSONALIZE` or a creation, modification or cancellation of a scenario is in progress
`ERR SCENE NOT CREATED`	The operation to create a new scenario can not be completed
`ERR READING DB`	The attempt to retrieve the plant returned an empty plant

dat.nguyendinh@vtd.com.vn

3.16 MODIFYSCENE

The MODIFYSCENE function allows the client to modify a scenario in the following parameters:

- name
- environment
- list of included SFs (or their state)
- list of included activators

Syntax of the MODIFYSCENE request

```
{
  "type":"request",
  "function":"modifyscene",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"14",
  "args":[
    {
      "idsf":168,
      "name":"Relax-2",
      "idambient":5,
      "sflist":[155,135,166],
      "activatorlist":[
        {
          "idsf":235,
          "sfetype":"SFE_Cmd_DownKey_SceneActivator",
          "value":"Execute"
        }
      ]
    }
  ],
  "params":[]
}
```

The `args` parameter contains an object described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	identification of the SF of the scenario to be modified	168
`name`	name to change	“Relax-2”
`idambient`	identification of the environment	5
`sflist`	SF identifier set. The SFEs captured by the scenario are given by this set.	155,135,166
`activatorlist`	Activator list. An activator is an SFE of a specific class of SFs: the SF_SceneActivator. Note that the availability of activators is not guaranteed for any plant; details on activators can be found in the SF specification.	see table

dat.nguyendinh@vtd.com.vn

Syntax of the MODIFYSCENE response

```
{
  "type":"response",
  "function":"modifyscene",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"14",
  "error":0,
  "result":[]
}
```

In case of an error (ie, a value other than `0` of the `error` parameter), the changes will be discarded. The `result` parameter does not contain any data.

The following are the possible return codes specific to the `error` parameter:

ERROR CODE	MEANING
ERR MALFORMED ARGS	Within the first object of the `args` array, one of the following fields is not present: • `idsf` (integer)
ERR DATA	System Function with id equal to `idsf` not found or does not belong to the scenarios domain or none of the optional parameters `name`, `idambient`, `sflist` and `activatorlist` has been provided in the `args` field.
ERR PERMISSION DENIED	The user does not have permission `SCENE PERSONALIZE` or a creation, modification or cancellation of a scenario is in progress
ERR SCENE NOT SAVED	System Function Element snapshot or scenario activators saving failed

dat.nguyendinh@vtd.com.vn

3.17 DELETESCENE

The DELETESCENE function allows the client to delete a previously created scenario.

Syntax of DELETESCENE request

```
{
  "type":"request",
  "function":"deletescene",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "msgid":"15",
  "args":[
    {
      "idsf":168
    }
  ],
  "params":[]
}
```

The `args` parameter contains an object described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`idsf`	identification of the scenario SF to be deleted	168

The `params` parameter contains no objects.

Syntax of the DELETESCENE response

```
{
  "type":"response",
  "function":"deletescene",
  "source":"e6fd3d010e2341d4",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"15",
  "error":0,
  "result":[]
}
```

Here follows the possible return codes related to the `error` parameter:

ERROR CODE	MEANING
`ERR MALFORMED ARGS`	Within the first object of the `args` array, one of the following fields is not present: • `idsf` (integer)
`ERR PERMISSION DENIED`	The user does not have not have permission `SCENE PERSONALIZE` or create/modify/delete scenario activity is in progress

dat.nguyendinh@vtd.com.vn

`ERR SCENE NOT DELETED`	An error occurred during the scenario deletion.

dat.nguyendinh@vtd.com.vn

3.18 RETRIEVELANGDICTIONARY

The RETRIEVELANGDICTIONARY allows the client to retrieve text dictionary.

Syntax of the RETRIEVELANGDICTIONARY request

```
{
  "type":"request",
  "function":"retrievelangdictionary",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "token":"dFvC395h6UsR",
  "msgid":"0",
  "args":[
    {
        "lang":"it"
    }
  ],
  "params":[]
}
```

The client sets the `lang` filed with one of the supported languages values (4). If the value is incorrect, an error is returned.

Syntax of the RETRIEVELANGDICTIONARY response

```
{
    "type":"response",
    "function":"retrievelangdictionary",
    "source":"e6fd3d010e2341d4",
    "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
    "msgid":"0",
    "error":"0",
    "result":[
        {
            "lang": "it",
            "hash": "DICTIONARY_HASH",
            "dictionary": {
                "12345": "Lampadario centrale",
                "9999": "Soggiorno",
                "3333": "Relax"
                }
        }
    ]
}
```

The dictionary will contains the texts for:

- environment names
- plant SF names (applications on ConfiguratorApp).
- gateway logs
- push notification texts

dat.nguyendinh@vtd.com.vn

Here follows the possible return codes related to the `error` parameter:

ERROR CODE	MEANING
`ERR MALFORMED ARGS`	Within the first object of the `args` array, one of the following fields is not present: • `lang` (string)
`ERR LANG NOT SUPPORTED`	Language not supported

dat.nguyendinh@vtd.com.vn

3.19 GETLOG

This call allows the client to ask the server for the event log.

Syntax of the GETLOG request

```
{
  "type":"request",
  "function":"getlog",
  "source":"e6fd3d010e2341d4",
  "token":"WjJkRkZGRkY=",
  "target":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "msgid":"10",
  "args":[
    {
      "levels":["alarm","info"],
      "categories":["info"],
      "from":1234567890,
      "to":1234567899,
      "count":50,
      "skip":2
    }
  ],
  "params":[]
}
```

The `args` parameter contains an object described by the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
`levels`	Severity level	See table
`categories`	categories	See table
`from`	Start Timestamp (older)	See Table
`to`	End Timestamp (most recent)	See table
`count`	Number of requested events (block)	50
`skip`	Block index	From 0 (first) to n (following)

The parameters `count` and `skip` can be used to facilitate the UI to display events.
Example: Assuming that on UI events are displayed from the most recent to the oldest, with `count` is possible to specify the number of events (block) displayed in the available graphics area (page) and with `skip` is possible to specify which page it is.
Executing multiple queries with constant `count` and incrementing `skip`, is possible to scroll from the most recent event to the oldest one.

The `params` parameter contains no objects.

dat.nguyendinh@vtd.com.vn

Values assumed by `levels` paramters:

VALUE	DESCRIPTION	EXAMPLE
"alarm" "error" "warning" "info" "debug" "all"	Each value applies the related filter	["all"] ["error","warning"]

Values assumed by the `categories` parameter:

VALUE	DESCRIPTION	EXAMPLE
"info" "maintenance" "upgrade" "fault" "all"	Each value applies the related filter	["all"] ["maintenance", "upgrade"]

Values assumed by the `from` and `to` parameters:

VALUE	DESCRIPTION	EXAMPLE

Unix time	Value in seconds from January 1, 1970	Rome 7 August 2018 12:00
		Unix time=1533636000;
		from=1533636000

Syntax of the GETLOG response

```
{
  "type":"response",
  "function":"getlog",
  "source":"8YVGVZdtjKpLq6DbnG7WQ9km",
  "target":"e6fd3d010e2341d4",
  "msgid":"10",
  "error":0,
  "result":[
    {
      "timestamp":1234567894,
      "level":"alarm",
      "role":"",
      "category":"info",
      "dictkey":"1234",
      "params":{
        "P1":"10",
        "P2":"On",
        "P3":"100"
      }
    },
    {
      "timestamp":1234567892,
```

dat.nguyendinh@vtd.com.vn

```
        "level":"info",
        "role":"",
        "category":"info",
        "dictkey":"4321",
        "params":{
          "P1":"On"
        }
      }
    ]
  }
```

The information returned in the `result` parameter is an array of objects each related to a single LOG event.
The order of the elements in the array is chronological starting from the most recent event.

Here follows the parameters of each object of the `result` parameter:

PARAMETER	DESCRIPTION	EXAMPLE
`timestamp`	Unix time	Value in seconds from January 1, 1970. It will be the client's task to display the date-time taking into account the timezone of the plant and with the format appropriate to the UI (dd / mm / yy, mm / dd / yy, 24h, am / pm ...). It will be greater than or equal to the value indicated in the `from` parameter of the request.
`level`	Severity level	The value belongs to one of the severity levels indicated in the request
`role`	Role of the user in the session	It is calculated by the server that inserts the client user role The value can be one of the following: "user": normal user "dev": developer user
`categories`	Categories	Value belongs to one of the categories indicated in the request
`dictkey`	Identifier of the event description string, used to translate in the language set in the App	"4321"
`params`	Any parameter associated with this event, identified by a key composed of the letter P and a progressive number starting from 1, followed by the string of the value	

Here follows the possible return codes related to the `error` parameter:

dat.nguyendinh@vtd.com.vn

ERROR CODES	MEANINGS
`ERR MALFORMED ARGS`	In the first object of the `args` array, one of the following fields is not present: • `levels` non-empty array of strings • `categories` non-empty array of strings • `from` integer without sign • `to` integer without sign • `count` integer • `skip` integer or the array `args` contains a number of elements other than 1.
`ERR DATA`	Category or severity level not allowed, fields `from` and `to` not coherent or field `count` exceeding the maximum allowed limit
`ERR SYSTEM BLOCK`	Failure of access validation request or log access request
`ERR PERMISSION DENIED`	Lack of permissions required for log access

dat.nguyendinh@vtd.com.vn

4 Appendix

4.1 Return codes

Return codes

```
#define IP_CONNECTOR_NO_ERR                           0

#define IP_CONNECTOR_ERR_MALFORMED_MESSAGE            1
#define IP_CONNECTOR_ERR_PERMISSION_DENIED            2
#define IP_CONNECTOR_ERR_INVALID_TARGET               3
#define IP_CONNECTOR_ERR_INVALID_SOURCE               4
#define IP_CONNECTOR_ERR_INVALID_TOKEN                5
#define IP_CONNECTOR_ERR_INVALID_TYPE                 6
#define IP_CONNECTOR_ERR_MALFORMED_ARGS               7
#define IP_CONNECTOR_ERR_UNKNOWN_FUNCTION             8
#define IP_CONNECTOR_ERR_UNIMPLEMENTED_FUNCTION       9
#define IP_CONNECTOR_ERR_DATA                         10
#define IP_CONNECTOR_ERR_ELEMENT_VALUE                11
#define IP_CONNECTOR_ERR_USER_NOT_FOUND               12
#define IP_CONNECTOR_ERR_INVALID_CLIENT_TAG           13
#define IP_CONNECTOR_ERR_INVALID_COMM_MODE            15
#define IP_CONNECTOR_ERR_SESSION                      16
#define IP_CONNECTOR_ERR_REMOVING_USER                17
#define IP_CONNECTOR_ERR_SCENE_NOT_CREATED            18
#define IP_CONNECTOR_ERR_SCENE_NOT_DELETED            19
#define IP_CONNECTOR_ERR_SCENE_NOT_SAVED              20
#define IP_CONNECTOR_ERR_INVALID_PWD                  21
#define IP_CONNECTOR_ERR_SYSTEM_BLOCK                 23
#define IP_CONNECTOR_ERR_SESSION_ALREADY_STARTED      24
#define IP_CONNECTOR_ERR_MALFORMED_PARAMS             25
#define IP_CONNECTOR_ERR_READING_DB                   27
#define IP_CONNECTOR_ERR_PLANT_NOT_CONFIGURED         28
#define IP_CONNECTOR_ERR_LANG_NOT_SUPPORTED           31
#define IP_CONNECTOR_ERR_SYSTEM_LOADING               34
#define IP_CONNECTOR_ERR_INVALID_CONTEXT              35
```

4.2 Supported languages

CODE	LANGUAGE
it	Italian
en	English
fr	French
de	German
es	Spanish
el	Greek
ru	Russian
zh	Chinese

dat.nguyendinh@vtd.com.vn

`he`	Hebrew
`pt`	Portuguese
`pl`	Polish
`tr`	Turkish
`ar`	Arabic
`sv`	Swedish
`nl`	Dutch

dat.nguyendinh@vtd.com.vn

References

[1] I. Fette and A. Melnikov. The WebSocket Protocol. RFC 6455, RFC Editor, http://www.rfc-editor.org/rfc/rfc6455.txt, December 2011.

[2] F. Yergeau. UTF-8, a transformation format of ISO 10646. RFC 3629, RFC Editor, http://www.rfc-editor.org/rfc/rfc3629.txt, November 2003.

[3] Vimar S.p.a. Modello System Functions per Automation Gateway. Specifica tecnica, Vimar S.p.a.

[4] R. Salz P. Leach, M. Mealling. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122, RFC Editor, http://www.rfc-editor.org/rfc/rfc4122.txt, July 2005.

[5] S. Cheshire and M. Krochmal. Multicast DNS. RFC 6762, RFC Editor, http://www.rfc-editor.org/rfc/rfc6762.txt, February 2013.

[6] S. Cheshire and M. Krochmal. DNS-Based Service Discovery. RFC 6763, RFC Editor, http://www.rfc-editor.org/rfc/rfc6763.txt, February 2013.

dat.nguyendinh@vtd.com.vn