RehabConnex Server Protocol Definition

Abstract

This document contains the definition of the *Rehab Connex* protocol and all commands known by the *Rehab Connex* Server up to version 0.7. There are also some example configuration messages and the corresponding answers from the server at the end of this document.

Reference

http://rehabconnex.zhdk.ch

Version

0.87 direct patches for configurations

0.7 access-control, auto-patches, new defintions for slots and devices (functions, relay)

0.6 push&pull-slots (direct communication push), pushpull-nodes

0.5 paths with values (nodename,nodeargument)
get j4 root.server.'rehabserver'.devices.device.slotsets.normalized.slotset.glove.outx

Send/Receive parameters

List of the parameters needed to communicate with *Rehab Connex* server:

Parameter name	Description
command	The command to be executed on the server or in case of a message received from the server the *return* command.
jobid	A custom defined job identifier to match the send message with the corresponding answer.
path or id	The path in the server structure where the command is executed. path node name only: root.server.clients.client.slotsets path with names: root.server.clients.'glove'.slotsets id: 2
arguments	Command arguments, see *Commands* for detail.

Commands

List of commands known to the *Rehab Connex* server.

Command name	Description
Structure Add nodes, add devices, add slotsets, add slot, remove root.server.clients root.server.clients clientx root.server.clients clientx.devices	

root.server.clients clientx.devices.devicey root.server.clients clientx.devices.devicey.slotsets …	
Add	Adds a new node to the server at the location specified in the parameter *path*. The nodetype is specified as the first *argument*.
adddevice add.device	Adds a new device to the server at the location specified in the parameter *path*.
addslotset add.slotset	Adds a slotset node to the slotsets at the location specified in the parameter *path*.
addslotsets add.slotset	Adds a slotsets node to the device at the location specified in the parameter *path*.
addslots add.slots	Adds a slots node to the slotset at the location specified in the parameter *path*.
addslot add.slot	Adds a slot to the slot specified in the parameter *path* with the name specified in the parameter *argument*.
insertdeviceinputslot Insert.deviceinputslot	Adds a device, slotset and slot with the device name specified in parameter *argument*.
insertinputslotat Insert.inputslotat	Adds a inputslot at the location specified in the parameter *path*.
remove	Removes the node specified in the parameter *path* form the server structure recursively. Remove works only if the client is the owner the object.
set/stream and get Set and get values from rehabconnex. Address: Global: - root.server.clients.clientx.slotsets.normalized.slots.outx Local: - this.slotsets.normalized.slots.outx	
get	Returns the object (client, device, slotset, ...) or property (id, parent, name, nodetype, ...) specified in the parameter *path*. get job1 root Following direct calls are possible! get j0 clients returns clients ids get j1 devices returns devices ids get j2 slots: returns slots ids get j3 slotsets: returns all slotsets get j4 patches: returns all patches special function get j5 structure
get.id	Returns the id of the node specified in the parameter *path*.
get.name	Returns the name of the node specified in the parameter *path*.
get.parent	Returns the parent of the node specified in the parameter *path*.

get.client	Returns the parent client (recursive up)
get.device	Returns the parent device of this node (id)
get.owner	Returns 'true' or 'false' if calls client is owner
get.slotset	Return the correct slotset
get jobid slots	Returns the slots
get jobId slots.input	Return the input slots
get jobID slots.output	Return the output slots
get.argument	Returns the argument of the node specified in the parameter *path*.
get.path	Returns the path form root to the node specified in the parameter *path*.
get.length	Returns the number of children of the node specified in the parameter *path*.
get.objects	Returns the id's of all child objects of the node specified in the parameter *path*. The delimiter is ",".
set	Sets the value of the node specified in the parameter *path* to the value of the *argument*.
stream	(not implemented yet)
push /pull Works different than set and get! It is a direct communication to a slot. Use #hashes to push and pull data! Only works on pushha	
push	Set a param to a pushpull-slot! push path|id#param value
pull	Get a param from a pushpull-slot pull path|id#param
patching Generate patches. Delete them with the simple remove command and the id of the patch	
patch	Creates a patch. A patch pointes from the output-slot to an input-slot. Every SET on this slot will be redirected to target slot as a SET or stream command! Patches are attachted to the origin (output) Patch j1 PATCH|ID PATCH|ID Special: patch a PushPull-Slot Patch j1 outputSlot PushPull-Slot > a patch.pushpull will be created.
patch.auto	Creates an auto patch processed by the system. The autopatches are stored at root.server.patchesauto and can be removed there. patch.auto j1 inputslot outputslot It is also possible to add the autopaches in the config! Also clients can add autopatches!

Example messages

Adding Clients

Client objects will be created autmatically.

Message to set the name of your client internally (server.clients.xyz)
send > set.name job2 this “glove”
return > reply job2 ok

Adding Devices, Slots

Message to add a device:
send > add job1 device
return > reply job1 ok 1234

Message to set the name of this new added device:
send > set job2 **this.**device.1234.name “glove”
return > reply job2 ok

Message to get the name of this new added device:
send > get job3 device.1234.name
return > reply job3 ok glove

// todo: ???
Message to set the internal name of this new added device:
send > set.name job2 device.1234 “glove”
return > reply job2 ok

Message to get the name of this new added device, but with wrong device id:
send > get job4 device.9999.name
return > reply job4 error object not found

Message to set the description of a device:
send > set job5 device.1234.desc “This is a glove.”
return > reply job5 ok

Message add a device with inputslots in one step:
send > insertdeviceinputslot job6 “gloveright”
return > reply job6 ok “path_of_the_slot”

Set/Get

Set and get arguments. Every node has a name and a argument/value. If you wanna change the name of the nodes, than use set.name|get.name

Base Structure:

1. root: (list){-1}
- 2. server: (server){-1}
-- 3. [name:RehabConnex (string){-1}]
-- 4. [version:0.5 (float){-1}]

-- 5. clients: (list){-1}
--- 6. client6:simpleclient (client){-1}
---- 7. [ip:192.168.1.36 (string){-1}]
---- 8. [port:59672 (string){-1}]
---- 9. devices: (list){-1}

Attention: No difference between path and id!

Structure: get|set JOBID PATH|ID

PATH:
- root.server.version
- root.server.clients.'*simpleclient '.ip*

local
- this.outx

ID:
- 4
- 6.name

Message to get the server name
```
send> get j1 root.server.name
return> reply j1 ok RehabConnex
```

Message to get the server version
```
send> get j1 root.server.version
return> reply j1 ok 0.3
```

Path/Id convertions

Message to get the path for this id
```
send> get.path j1 7
return> reply j1 ok root.server.clients.client6.ip
```

Message to get the id for this path
```
send> get.id j1 root.server.clients.client6.ip
return> reply j1 ok 7
```

Searching (Base functions)

Message to get the clients list
```
send> get j1 clients
return> reply j1 ok 1,6
```

Message to get the clients name of the first
```
send> get .name j2 6
return> reply j2 ok rehabdeviceabc
```
Message to get the clients name of the first
```
send> get  j2 6
return> reply j2 ok REHABCLIENTXYZ
```

Message to get a device list
```
send> get j1 devices
return> reply j1 ok 5,10,20
```

Message to get the device name of the first
```
send> get .name j2 5
return> reply j2 ok glovex
```

Message to get a slotset list
```
send> get j1 slotsets
return> reply j1 ok 7,11,21
```

Message to get a slots list
```
send> get j1 slots
return> reply j1 ok 9,14,21
```

Message to get the slot name of the first
```
send> get .name j2 9
return> reply j2 ok x
```