Table of Contents

About This Document

This document is a VX LWCP protocol reference.

Author: Nikita Petrovs.

Change Log

18/03/2010 16:23:12

Initial version.

Protocol Version

Protocol version is 0.9.7.

General Notes About the VX LWCP Protocol

VX LWCP protocol is an extension over standard LWCP protocol. It is case-sensitive as opposed to the standard LWCP protocol.

Property types are the same as in standard LWCP protocol:

- String
- Name (aka enumerated value. Identical to string, but without quotes)
- Number
- List

There is a pseudo type of enumerated boolean which holds two values of TRUE and FALSE.

All the lists must maintain element order. It is advised that “indi” property response to “get” request maintain the order of properties as they were sent in the “get” request. It is also advised that all events maintain the property order as specified in this document.

Like the standard LWCP this version have the same syntax:

```
operation object.subobject#SOBJ_ID property1, property2=value2
```

Protocol has two standard operations derived from the generic version: “get” and “set”. Almost all properties in this protocol are gettable, but only few are settable.

Syntax for mentioned above operations is:

```
get object property
```

```
set object property=value
```

The reply for the “get” operations always should be “indi”. Nonexistent properties should be ignored.

Several properties can change their type depending on the system state. For example “show_name” can be NULL (enumerated value) though it's real type is string.

Server responses

Server must reply using one of the following operations:

- ack – mostly used to report an error but also used as status notification for example for “login”, “sub” and “unsub” operations
- indi – mostly used to return the values requested by “get” operation
- event – asynchronous notification of the system state change
- pong – reply to ping operation
- update – notification of the record change

Error reporting is done using “ack” operation. For example this is notification of a misspelled login request:

```
ack cc $status=ERR $msg="Property 'user' or 'password' is missing."
```

Predefined properties for error reporting are “$status”, “$code” and “$msg”. “$status” is a mandatory property but “$code” and “$msg” are optional though it is advices to use “$msg” property.

VX LWCP Objects

VX LWCP protocol has 7 different types of objects which are “cc”, “studio”, “line”, “book” and “log”. Some of those are sub objects meaning that you'll have to use them in “object.subobject” fashion.

Object “cc”

Object “cc” is a root object of the protocol. It holds the server information properties as well as list of available studios.

Properties

studio_list

This is a read-only list property containing the list of available studios. Every list element is another list of two elements – studio ID and studio name. “ID” is a number and “name” is a string. List can be empty meaning that no studios were configured.

```
get cc studio_list
indi cc studio_list=[[1, "Studio 1"], [2, "Studio 2"], [3, "Studio 3"]]
```

date

This is a read-only string property mostly used for the VX phones to set their time cause the don't have any RTC. Type of it is string and format is “YYYY-MM-DDThh:mm:ss” conforming to ISO 8601.

```
get cc date
indi cc date="2010-03-18T17:04:33"
```

server_id

This is a read-only string property containing server identification string. It is accessible without logging in. Server developers should decide by themselves what value will it have.

```
get cc server_id
indi cc server_id="Telos VX"
```

server_version

This is a read-only string property containing server version string. It is accessible without logging in.

```
get cc server_version
indi cc server_version="0.9.7"
```

server_caps

This is a read-only string property containing server capabilities string. It is accessible without logging in.

The initial capabilities set is “b”.

```
get cc server_caps
indi cc server_caps="b"
```

lwcp_version

This is a read-only integer property containing VX LWCP version server is using and is of number type. It is accessible without logging in.

Initial value is 1.

```
get cc lwcp_version
indi cc lwcp_version=1
```

mode

This is a write-only enumerated property that tells server of a current client mode. Depending on the mode some operations may have different behavior, like taking the next studio line.

It must be one of two possible values: “TALENT” or “PRODUCER” without quotes.

```
set cc mode=TALENT
```

Operations

login

This is the essential operation for further use of the protocol. If client is not logged in then it only has access to cc object's for properties “server_id”, “server_version”, “server_caps”, “lwcp_version” as well as it can use “ping” operation.

Login is an overloaded operation with two variants.

First is a standard way to login into the system:

```
login cc user="some_username" password="some_password"
```

Both “user” and “passwords” are mandatory and both are of a string type.

Second one is meant for the flash meters applet:

```
login cc sessionid="web_session_id"
```

“sessionid” is an ID of an active web session stored in cookie “telos_session_id”.

Server reply must be either

```
ack cc logged=FALSE
```

if login fails or

```
ack cc logged=TRUE
```

if login succeeds.

ping

This is a very simple operation used for one thing – to test if VX server is still alive. It is accessible without logging in. Syntax is:

```
ping cc
```

Server answer must be:

```
pong cc
```

Object “studio”

This is the main object for the configured studio. It has various properties as well as several sub objects like “line”, “book” and “log”.

Properties

id

This is a read-only integer property containing studio ID. It corresponds to studio's position in the “studio_list” of the “cc” object. Studio is selected by ID or by name using the “select” operation of the “studio” object.

```
get studio id
indi studio id=1
```

name

This is a read-only string property containing studio name. Studio is selected by ID or by name using the “select” operation.

```
get studio name
indi studio name="Studio 1"
```

show_list

This is a read-only list property containing the list of available shows for this studio. Every list element is another list of two elements – show ID and show name. “ID” is a number and “name” is a string. List can be empty meaning that no shows are available.

```
get studio show_list
indi studio show_list=[[1, "Show 1"], [2, "Show 2"], [3, "Test 1"]]
```

show_id

This is a read-only integer property containing currently selected show ID. It corresponds to the show's position in the “show_list” property of the “studio” object. Show is selected by ID or by name using the “select_show” operation of the “studio” object.

```
get studio show_id
indi studio show_id=1
```

show_name

This is a read-only string property containing the name of currently selected show. Studio is selected by ID or by name using the “select” operation.

```
get studio show_name
indi studio show_name="Show 1"
```

line_list

This is a read-only list property containing the list of available lines.

List elements in exact same order are the following line properties: state, callstate, name, local, remote, hybrid, time, comment and direction. List element can be NULL instead of another list meaning that an empty line is configured with following default properties: [NONE, NONE, “”, “”, NULL, 0, NULL, “”, NONE]

```
get studio line_list
indi studio line_list=[[IDLE, IDLE, "Main-Studio", "10", NULL, 0, NULL, "",
  NONE], [IDLE, IDLE, "Main-Studio", "10", NULL, 0, NULL, "", NONE], [IDLE,
  IDLE, "Main-Studio", "10", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "Line
  20", "20", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "67-614-448",
  "67614448", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "67-614-449",
  "67614449", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "Cisco 40", "40", NULL,
  0, NULL, "", NONE], [IDLE, IDLE, "Cisco 80", "80", NULL, 0, NULL, "",
  NONE], NULL, [IDLE, IDLE, "VIP", "88", NULL, 1, NULL, "", NONE], [IDLE,
  IDLE, "NEWS", "89", NULL, 2, NULL, "", NONE], [IDLE, IDLE, "HOT-LINE",
  "90", NULL, 3, NULL, "", NONE]]
```

num_lines

This is a read-only integer property containing the total number of lines in the selected studio.

```
get studio num_lines
indi studio num_lines=12
```

hybrid_list

This is a read-only list property containing the list of configured hybrids in this studio. List elements are strings with hybrid names.

```
get studio hybrid_list
indi studio hybrid_list=["Fixed 1", "Fixed 2", "Fixed 3", "Fixed 4", "S1-
  Selectable 1", "S1-Selectable 2", "Selectable 6", "Selectable 7"]
```

num_hybrids

This is a read-only integer property containing the total number of configured hybrids (fixed + selectable) in this studio.

```
get studio num_hybrids
indi studio num_hybrids=8
```

num_hyb_fixed

This is a read-only integer property containing the number of configured fixed hybrids in this studio. Fixed hybrids always go before the selectable in the list. Thus if we have 3 hybrids and 2 of them are fixed, then list will be [F, F, S]. Knowing the total number of hybrids and the number of fixed ones it is easy to get the number of selectables but dividing fixed count from total count.

```
get studio num_hyb_fixed
indi studio num_hyb_fixed=4
```

next

This is a read-only integer property which holds the ID number of the line that will be taken when take next operation is used. This is the queue for “TALENT” mode.

```
get studio next
indi studio next=0
```

pnext

This is a read-only integer property which holds the ID number of the line that will be taken when take next operation is used. This is the queue for “PRODUCER” mode.

```
get studio pnext
indi studio pnext=0
```

busy_all

This is a read-only enumerated boolean property that tells if the studio is in “busy all” state.

```
get studio busy_all
indi studio busy_all=FALSE
```

mute

This is a read-only enumerated boolean property that tells if the “mute” flag was set in this studio.

```
get studio mute
```

```
indi studio mute=FALSE
```

show_locked

This is a read-only enumerated boolean property that tells if the selected show is locked in this studio.

```
get studio show_locked
indi studio show_locked=FALSE
```

auto_answer

This is a read-only enumerated boolean property that tells if the “auto_answer” flag was set in this studio.

```
get studio auto_answer
indi studio auto_answer=FALSE
```

Operations

select

This is the initial operation required to start working with certain studio. After selecting the studio client will receive all state change events from it. As well as it will have access to studio specific (and also to the studio sub object's) operations. Studio can be selected either by ID or by name.

On successful selection an event should be sent back to the client with following properties: id, name, show_id, show_name, next, pnext, busy_all, num_lines, num_hybrids, num_hyb_fixed, mute, show_locked, auto_answer, line_list.

Studio can be deselected choosing the studio with id=0 or with name=NULL.

Has one integer parameter “id”.

```
select studio id = 1
event studio id=1, name="Studio 1", show_id=1, show_name="Show 1", next=0,
  pnext=0, busy_all=FALSE, num_lines=12, num_hybrids=8, num_hyb_fixed=4,
  mute=FALSE, show_locked=FALSE, auto_answer=FALSE, line_list=[[IDLE, IDLE,
  "Main-Studio", "10", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "Main-Studio",
  "10", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "Main-Studio", "10", NULL, 0,
  NULL, "", NONE], [IDLE, IDLE, "Line 20", "20", NULL, 0, NULL, "", NONE],
  [IDLE, IDLE, "67-614-448", "67614448", NULL, 0, NULL, "", NONE], [IDLE,
  IDLE, "67-614-449", "67614449", NULL, 0, NULL, "", NONE], [IDLE, IDLE,
  "Cisco 40", "40", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "Cisco 80", "80",
  NULL, 0, NULL, "", NONE], NULL, [IDLE, IDLE, "VIP", "88", NULL, 1, NULL,
  "", NONE], [IDLE, IDLE, "NEWS", "89", NULL, 2, NULL, "", NONE], [IDLE,
  IDLE, "HOT-LINE", "90", NULL, 3, NULL, "", NONE]]
```

select_show

This operation allows to change the active show by ID or by it's name. Has one integer parameter “id”.

On successful selection an event should be sent back to the client with following properties: show_id, show_name, next, pnext, num_lines, show_locked and line_list.

```
select_show studio id = 1
event studio show_id=1, show_name="Show 1", next=0, pnext=0, num_lines=12,
  show_locked=FALSE, line_list=[[IDLE, IDLE, "Main-Studio", "10", NULL, 0,
  NULL, "", NONE], [IDLE, IDLE, "Main-Studio", "10", NULL, 0, NULL, "",
  NONE], [IDLE, IDLE, "Main-Studio", "10", NULL, 0, NULL, "", NONE], [IDLE,
  IDLE, "Line 20", "20", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "67-614-
  448", "67614448", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "67-614-449",
  "67614449", NULL, 0, NULL, "", NONE], [IDLE, IDLE, "Cisco 40", "40", NULL,
  0, NULL, "", NONE], [IDLE, IDLE, "Cisco 80", "80", NULL, 0, NULL, "",
  NONE], NULL, [IDLE, IDLE, "VIP", "88", NULL, 1, NULL, "", NONE], [IDLE,
  IDLE, "NEWS", "89", NULL, 2, NULL, "", NONE], [IDLE, IDLE, "HOT-LINE",
  "90", NULL, 3, NULL, "", NONE]]
```

im

Operation is used to send instant messaged inside the studio. Has two string parameters “from” and “message”.

```
im studio from="Telos VX", message="Hello World!"
event studio from="Telos VX", message="Hello World!"
```

busy_all

Operation is used to put the studio in busy all state meaning that the lines that have been configured to be able to be busy will change their state to BUSY. Has one enumerated boolean parameter “state”.

Line will change it's state only if it's idle. If there's an active call on it then the state will change after that call is dropped.

```
busy_all studio state=TRUE
event studio busy_all=TRUE
event studio.line#1 state=BUSY, callstate=IDLE
event studio.line#2 state=BUSY, callstate=IDLE
event studio.line#4 state=BUSY, callstate=IDLE
```

drop

Operation allows to drop the calls that reside on a certain hybrid. Has one integer parameter “hybrid”.

```
drop studio hybrid = 5
event studio.line#1 state=IDLE, callstate=TERMINATED, hybrid=0, time=NULL
event studio.line#1 state=IDLE, callstate=IDLE, remote=NULL, direction=NONE,
  hybrid=0, time=NULL
```

hold

Operation allows to put the calls that reside on a certain hybrid on hold. Has one integer parameter “hybrid”.

```
hold studio hybrid = 5
event studio.line#1 state=ON_HOLD, callstate=ESTABLISHED, hybrid=0, time=0
```

```
event studio next=1, pnext=0
```

delay_dump

Not implemented.

Object "line"

Properties

state

This is a read-only enumerated property that holds the current state of the line. Can be one of the following values:

- NONE
- IDLE
- USED_EW
- SEIZED
- SEIZED_EW
- ON_HANDSET
- ON_HANDSET_EW
- RINGING_IN
- ON_AIR
- ON_AIR_LOCKED
- ON_HOLD
- ON_HOLD_READY
- BUSY

```
get studio.line#1 state
indi studio.line#1 state=ON_AIR
```

callstate

This is a read-only enumerated property that holds the current state of the call that resides on this line. Can be one of the following values:

- NONE
- IDLE
- DIALING
- RINGING_OUT
- RINGING_IN

- ACCEPTING
- ESTABLISHED
- TERMINATED

```
get studio.line#1 callstate
indi studio.line#1 callstate=ESTABLISHED
```

name

This is a read-only string property that holds the name of the line.

```
get studio.line#1 name
indi studio.line#1 name="Main-Studio"
```

local

This is a read-only string property that holds the local number of the line dialing which the call will be assigned to this line.

```
get studio.line#1 local
indi studio.line#1 local="10"
```

remote

This is a read-only string property that holds the remote number which has called this line.

```
get studio.line#1 remote
indi studio.line#1 remote="28"
```

hybrid

This is a read-only integer property that holds the hybrid currently assigned to this line.

```
get studio.line#1 hybrid
indi studio.line#1 hybrid=5
```

time

This is a read-only integer property that holds the time that passed from the beginning of the call in seconds. It resets when the line state is changed to a different one.

```
get studio.line#1 time
indi studio.line#1 time=123
```

comment

This is a read-write string property that holds the comment assigned to this line. Comment will reset when the call is dropped. Also comment can only be set when line state is not idle, when there is an active call.

```
set studio.line#1 comment = "This is a comment."
event studio.line#1 comment="This is a comment."
```

```
get studio.line#1 comment
indi studio.line#1 comment="This is a comment."
```

direction

This is a read-only enumerated property that holds the direction of the call. Can be on of three values: NONE, OUTGOING, INCOMING.

```
get studio.line#1 direction
indi studio.line#1 direction=OUTGOING
```

ssid and rport

These integer properties are sent only once when the call is put on handset and are unreadable after. “ssid” is a ID of the stream client should listen to and “rport” is the port to which client should send it's audio.

```
call studio.line#1 number="28", handset=TRUE
event studio.line#1 state=SEIZED, callstate=IDLE, hybrid=0, time=NULL
event studio.line#1 state=ON_HANDSET, callstate=DIALING, cause=100,
  remote="28", direction=OUTGOING, hybrid=0, time=0
event studio.line#1 state=ON_HANDSET, callstate=RINGING_OUT, cause=180,
  hybrid=0, time=0
event studio.line#1 state=ON_HANDSET, callstate=ESTABLISHED, cause=200,
  ssid=0, rport=62026, hybrid=0, time=0
```

cause

This integer property is a SIP response code sent in an event when the call state changes.

```
event studio.line#1 state=ON_HANDSET, callstate=ESTABLISHED, cause=200,
  ssid=0, rport=62026, hybrid=0, time=0
```

Operations

seize

Operation reserves the line for the client that no one else can call from it. Has no parameters.

```
seize studio.line#1
event studio.line#1 state=SEIZED, callstate=IDLE, hybrid=0, time=NULL
```

call

Operation creates a call to a specified number putting it either on certain hybrid or on handset. Has three parameters: string “number”, enumerated boolean “handset”, integer “hybrid”, integer “port”. “number” is a required parameter. “handset” determines which one of “port” and “hybrid” will have effect. If “handset” is TRUE then “hybrid” will have effect otherwise “port” will.

```
call studio.line#1 number="28"
event studio.line#1 state=ON_AIR, callstate=DIALING, cause=100, remote="28",
  direction=OUTGOING, hybrid=5, time=0
```

```
event studio.line#1 state=ON_AIR, callstate=RINGING_OUT, cause=180,
  hybrid=5, time=0
```

take

Operation takes the call either on air on certain hybrid or on handset. When no line index is specified then operation takes the "next" call.

```
take studio.line#1 handset=FALSE, hybrid=6

event studio next=0, pnext=0

event studio.line#1 state=ON_AIR, callstate=ACCEPTING, cause=200, hybrid=6,
  time=30

event studio.line#1 state=ON_AIR, callstate=ESTABLISHED, cause=200,
  hybrid=6, time=0


take studio.line#1 handset=TRUE

event studio next=0, pnext=0

event studio.line#1 state=ON_HANDSET, callstate=ACCEPTING, cause=200,
  hybrid=0, time=11

event studio.line#1 state=ON_HANDSET, callstate=ESTABLISHED, cause=200,
  ssid=0, rport=62010, hybrid=0, time=0


take studio.line

event studio next=0, pnext=0

event studio.line#1 state=ON_AIR, callstate=ACCEPTING, cause=200, hybrid=5,
  time=4

event studio.line#1 state=ON_AIR, callstate=ESTABLISHED, cause=200,
  hybrid=5, time=0
```

drop

Operation drops the call from this line. Has no parameters.

```
drop studio.line#1

event studio.line#1 state=IDLE, callstate=TERMINATED, hybrid=0, time=NULL

event studio.line#1 state=IDLE, callstate=IDLE, remote=NULL, direction=NONE,
  hybrid=0, time=NULL
```

lock

Operation locks the call on this line. Line state must be ON_AIR for this operation to succeed. Has no parameters.

```
lock studio.line#1

event studio.line#1 state=ON_AIR_LOCKED, callstate=ESTABLISHED, hybrid=5,
  time=7
```

unlock

Operation unlocks the call on this line. Line state must be ON_AIR_LOCKED for this operation to succeed. Has no parameters.

```
unlock studio.line#1
event studio.line#1 state=ON_AIR, callstate=ESTABLISHED, hybrid=5, time=41
```

hold

Operation puts the line on hold merging the caller with configured in studio on hold channel hybrid. Has one enumerated boolean parameter “ready” which determines which state to put this line to between ON_HOLD and ON_HOLD_READY.

```
hold studio.line#1 ready=TRUE
event studio.line#1 state=ON_HOLD_READY, callstate=ESTABLISHED, hybrid=0,
  time=0
event studio next=1, pnext=0
hold studio.line#1 ready=FALSE
event studio.line#1 state=ON_HOLD, callstate=ESTABLISHED, hybrid=0, time=29
event studio next=1, pnext=0
```

dtmf

Not implemented.

raise

Operation raises the priority of the line in the next queue. Only one line can have extra priority meaning that raising another line will put this one back where it was. Has no parameters.

```
raise studio.line#1
event studio next=1, pnext=0
```

sub

Operation used to subscribe to receive hybrid meters. Has two void parameters “meters” and “refresh”.

```
sub studio.line#1 meters
ack $status=OK
indi studio.line#1 meters=[0,-37,0,25,-34,-35,-470,-504,-266,-238,0]
indi studio.line#1 meters=[0,-37,0,25,-34,-35,-609,-643,-266,-238,0]
indi studio.line#1 meters=[0,-37,0,25,-34,-35,-749,-783,-266,-238,0]
```

unsub

Operation used to unsubscribe from receiving hybrid meters. Has no parameters.

```
unsub studio.line#1 meters
ack $status=OK
```

```
unsub studio.line#1 meters
ack $status=OK $msg="Not subscribed"
```

Object “book”

Properties

count

This is a read-only integer property containing total record count of the address book in this studio.

```
get studio.book count
indi studio.book count=17
```

list

This is a read-only list property containing the actual address book list. List element is another list of 3 elements of types integer, string and string. First one is ID of the record, second is name and third is number.

```
get studio.book list
indi studio.book range=[1,17], list=[[1,"Phone1","1@192.168.0.24"],[2,"show
  number long string 1","1"],[3,"show number long string 2","2"],[4,"show
  number long string 3","3"],[5,"show number long string 4","4"],[6,"show
  number long string 5","5"],[7,"show number long string 6","6"],[8,"show
  number long string 7","7"],[9,"show number long string 8","8"],[10,"show
  number long string 9","9"],[11,"show number long string 10","10"],
  [12,"show number long string 11","11"],[13,"show number long string
  12","12"],[14,"show number long string 13","13"],[15,"show number long
  string 14","14"],[16,"show number long string 15","15"],[17,"show number
  long string 16","16"]]
```

range

This is an additional property for the address book list request which tells the server the starting record of the request and how many records to return. Type of the property is list of two integer elements. First element is starting record and second element is record count.

```
get studio.book list, range=[1,4]
indi studio.book range=[1,4], list=[[1,"Phone1","1@192.168.0.24"],[2,"show
  number long string 1","1"],[3,"show number long string 2","2"],[4,"show
  number long string 3","3"]]
```

Operations

add

Adds a new record to the address book. Has three parameters: “type”, “name” and “number”. “type” enumerated and can be GLOBAL, STUDIO or SHOW specifying the record scope. Other two are strings.

```
add studio.book name="New Record", number="123"
```

```
update studio.book#18 type=INSERT
```

set

Operation can change any of three available properties of the record.

```
set studio.book#18 name="Changed Record"
update studio.book#18 type=UPDATE
```

del

Operation deletes the record from address book. Has no parameters.

```
del studio.book#18
update studio.book#18 type=DELETE
```

Object "log"

Properties

count

This is a read-only integer property containing total record count of the call history in this studio.

```
get studio.log count
indi studio.log count=100
```

list

This is a read-only list property containing the actual call history list. List element is another list of 5 elements of types integer, integer, integer, string and string. First one is the starting time of the call, second is the duration of the call, third is the type of the call (0 – outgoing, 1 – incoming), fourth is the local number called and the last one is the remote number.

```
get studio.log list, range=[0,5]
indi studio.log range=[1,5],
  list=[[1267557538,0,0,"10@192.168.0.9","20@192.168.0.32"],
  [1267557471,0,0,"40@192.168.0.9","28@192.168.0.23"],
  [1267552599,0,0,"40@192.168.0.9","28@192.168.0.23"],
  [1267552464,0,0,"40@192.168.0.9","28@192.168.0.23"],
  [1267107788,1,0,"1@192.168.0.9","28@192.168.0.23"]]
```

range

This is an additional property for the call history list request which tells the server the starting record of the request and how many records to return. Type of the property is list of two integer elements. First element is starting record and second element is record count.

```
get studio.log list, range=[4,1]
indi studio.log range=[4,1],
  list=[[1267552464,0,0,"40@192.168.0.9","28@192.168.0.23"]]
```

Examples

Address book examples

```
-> get studio.book list{=[id1, id2, id3, ...]} {, range=[start,count]} {,
  count}
<- indi studio.book range=[start,count], list=[[ID,"Name","Number"], ...]
# no range returned if requested IDs in list param (list=[id1, id2,
  id3, ...])
OR
-> get studio.book#ID type, name, number
<- indi studio.book#ID type=TYPE, name="Name", number="Number"



# at least one param must be set
-> set studio.book#ID type = TYPE, name = "New Name", number = "New Number"
<- update studio.book#ID type=UPDATE

# 'name' and 'number' must be set
-> add studio.book#ID type = TYPE, name = "Name", number = "Number"
<- update studio.book#ID type=INSERT

-> del studio.book#ID
<- update studio.book#ID type=DELETE



# TYPE (enum) =
# {GLOBAL,
# STUDIO,
# SHOW}

get studio.book count
indi studio.book count=7
get studio.book list
indi studio.book range=[1,7], list=[[1,"aa_bb","11111"],[2,"bbbbb","22222"],
  [4,"ccccc","3333333"],[5,"ddddd","444444"],[6,"show name","123"],[7,"new
  entry","1111"],[8,"one more entry","444"]]
get studio.book list, range=[1,4]
indi studio.book range=[1,4], list=[[1,"aa_bb","11111"],[2,"bbbbb","22222"],
```

```
  [4,"ccccc","3333333"],[5,"ddddd","444444"]]
get studio.book list, range=[3,2]
indi studio.book range=[3,2], list=[[4,"ccccc","3333333"],
  [5,"ddddd","444444"]]
get studio.book#2 type, name, number
indi studio.book#2 type=STUDIO, name="bbbbb", number="22222"
set studio.book#2 name="My Name", number="1234567"
update studio.book#2 type=UPDATE
get studio.book#2 name, number
indi studio.book#2 name="My Name", number="1234567"
get studio.book list, range=[2,1]
indi studio.book range=[2,1], list=[[2,"My Name","1234567"]]
get studio.book#3 type, name, number
ack studio.book#3 $status=ERR $msg="Address book record with id 3 wasn't
  found."
get studio.book list=[1,2,3]
indi studio.book list=[[1,"aa_bb","11111"],[2,"My Name","1234567"]]
del studio.book#2
update studio.book#2 type=DELETE
get studio.book#2 name, number
ack studio.book#2 $status=ERR $msg="Address book record with id 2 wasn't
  found."
get studio.book list=[2]
indi studio.book list=[]
add studio.book type=SHOW, name="Me", number="333"
update studio.book#9 type=INSERT
get studio.book#9 type, name, number
indi studio.book#9 type=SHOW, name="Me", number="333"
set studio.book#9 number="123"
update studio.book#9 type=UPDATE
get studio.book#9 type, name, number
indi studio.book#9 type=SHOW, name="Me", number="123"
set studio.book#9 type=STUDIO
update studio.book#9 type=UPDATE
get studio.book#9 type, name, number
indi studio.book#9 type=STUDIO, name="Me", number="123"
```