API Stored - media DB service

The database is accessed via the XML-RPC protocol. To start working with the database, you need to open a session, all subsequent calls and identifiers will be processed by the server in the context of this session.

Opening a session

The session is opened by calling the following function:

Open_session

opens a session with the server

```
parameters - none
return value - int session id or error code (negative number) in case of
error
```

Closing a session

After finishing working with the database, the session should be closed.

Close_session

closes a previously opened session

```
parameter 0 - int - session identifier
return value "Res" - int - 0 if successful, or an error code (negative
number)
```

Session maintenance

To protect against incorrectly working applications, as well as, for example, loss of network connection, the server automatically closes the session after 20 seconds of inactivity in the context of this session. Therefore, if the session needs to be kept open, and there is no need to send any meaningful commands for a long time, you need to use a special function that keeps the session “warm”:

keep_session

prevents the session from being closed due to timeout

```
parameter 0 - int - session identifier
return value "Res" - int - 0 if successful, or an error code (negative
number)
```

Working with a media object

To work with any media object (reading attributes, changing attributes ...), it must be opened, after successful opening, the identifier of the open media object (descriptor) is passed to all functions for working with media objects.

open_link_ns

opens a media object

```
parameter 0 - session identifier
parameter 1 - string - namespace identifier
parameter 2 - string - object identifier in the selected namespace
return value "Res" - int - 0 if successful, or an error code (negative
number)
```

After finishing working with the media object, you need to close the descriptor:

close_link

closes the media object

```
parameter 0 - int - session identifier
parameter 1 - int - descriptor
return value "Res" - int - 0 if successful, or an error code (negative
number)
```

read_link

reads the attributes of a media object

```
parameter 0 - int - session identifier
parameter 1 - int - descriptor
return value is a structure containing the following fields:
    "Res" - int - 0 if successful, or an error code (negative number)
```

```
    "Inf" - structure link_info_t
```

list_folders

gets a list of folders

```
parameter 0 - int - session identifier
parameter 1 - int - should be -1
return value is a structure containing the following fields:
    "Res" - int - 0 if successful, or an error code (negative number)
    "Lst" - structure folder_list_t
```

list_links

gets a list of media objects in the folder. Either the complete list, or only changed objects from a certain revision of the base.  The change counter values are not tied to real time.

```
parameter 0 - int - session identifier
parameter 1 - int - the value of the counter of changes in the media base,
relative to which you need to get a list of changed assets. To get a list of
all assets in a folder use "-1".
parameter 2 - string - folder name

return value is a structure containing the following fields:
    "Res" - int - 0 if successful, or an error code (negative number)
    "Lst" - structure link_list_t
```

Description of structures and types

field_list

an array, each element of which is a structure containing the following fields

```
"N" - string - field name
"V" - string - field value
```

link_info_t

the structure describing the media object contains the following fields.

```
"Fin" - int - the originsl source path where the object was imported
"Fld" - string - the path where the object is located
"Id" - int - media object identifier
```

```
"Lf" - field_list - media object attributes
"Links" - int - the number of media objects linking to this material
"Nf" - field_list - material attributes
"Nid" - int - unique identifier of the Node ID
"Nstamp" - int - Node stamp
"Stamp" - int - Media object Stamp
"Dt" - double is an optional field. If it is present, then it means the time
of automatic deletion of the media object.
```

folder_list_t

the structure containing the list of folders contains the following fields.

```
"Full" - int - reserved
"Stamp" - int - reserved
"Folders" - array <string> - folder names
"Deleted" - array <string> - reserved
```

link_list_t

the structure describing the list of media objects contains the fields:

```
"Full" - int - 0 in the case when the "links" and "deleted" fields contain
the difference relative to the change counter specified in parameter 1 of
the list_links function. If "-1" is passed to the list_links function in
parameter 1, then "full" is guaranteed not equal to 0. However, if a value
other than "-1" is passed, "full" may still not be equal to 0, so when
processing the returned values □□must be checked for this flag.
"Stamp" - int - the current value of the revision counter
"Links" - array <link_info_t> - list of assets in the folder. If "full" is
equal to 0 - the list of changed or added assets to the folder, if "full" is
not equal to 0, then the complete list of assets.
"Deleted" - array <int> - list of link_id of deleted assets. If "full" is
not equal to "0" the list must be empty.
```

Attributes

For media objects and materials, there is a list of attributes that have a predefined value. To optimize the network traffic, each of these attributes is assigned a default value, if the attribute value matches the default value, then this attribute will be excluded from the list during transmission.

All attribute values are strings, if the attribute had a numerical value, for example, the duration of a clip, it must be converted to a numeric value. The following lists indicate boolean attribute types

Material attribute values

```
  "orig_in_point" - int - the entry point of the source from which it was
digitized
                          material. The default is 0
  "orig_type" - int - source type code from which the material was
digitized.
                      It can take the following values:
                        0 - unknown
                        1 - VTR
                        2 - LIVE
                        3 - FILE
                      default value is 0
  "orig_name" - string - name of the source from which the material was
digitized,
                         the default is ""
  "rec_time" - double - the time when the material was digitized (in Julian
days),
                        default value is 0
  "frames" - int - total duration in frames of the material,
                   default value is 0
  "frame_time" - double - duration in seconds of one frame,
                  default is 0.04
  "mime_type" - int - material type code, can take the following values:
                        0 - unknown
                        1 - Audio
                        2 - Video
                        3 - AudioVideo
                        4 - StillImage
                        5 - ColorMatte
                        6 - Text
                        7 - Complex
                        8 - Tape
                        9 - Stream
                      default is 3
  "src_name" - string - the name of the medium on which the material is
located,
                        the default is ""
  "src_type" - int - the code of the media type on which the material is
located,
                     can take the following values:
                       0 - unknown
                       1 - TapeBSP
                       2 - TapeDV
                       3 - TapeIMX
                       4 - Digital
                       5 - TapeVHS
                      default is 4
```

Media object attribute values

```
"Comment" - string - comment, default value - ""
"Title" - string - title, default value - ""
"First_frame" - int - entry point, default value - 0
"Last_frame" - int - exit point, default value - 0
```

Error codes

ERR_NOERR	0	No error
ERR_UNK	-1	Unknown error
ERR_SESS	-2	Invalid session descriptor
ERR_LD	-3	Invalid link descriptor
ERR_NFND	-4	Object not found
ERR_DUPID	-5	Operation would duplicate some id
ERR_INV	-6	Invalid parameter
ERR_FLD	-7	No such folder
ERR_BUSY	-8	Object in use now
ERR_PRF	-9	No such profile
ERR_CONT	-10	Content not found
ERR_AGAIN	-11	Request not returned yet
ERR_PERM	-12	Permission denied
ERR_BADOP	-13	Invalid operation
ERR_NOBRV	-14	No BRV
ERR_NOARC	-15	No ARC
ERR_HASARC	-16	Has ARC
ERR_HASBRV	-17	Has BRV
ERR_OP_NOLIC	-18	Operation Not Licensed
ERR_CONN_NO_LIC	-19	Connection not Licensed
ERR_CONTINUE	-20	Background operation in progress

Examples

Examples of pseudocode for checking the presence of a clip in the database and reading its attributes:

```
int sess = open_session ();
XmlRpcValue val = open_link_ns (sess, "title", "Title value");

int ld = val ["res"];
if (ld <0) {
  if (ld == ERR_NFND) {
    // Not found
  } else {
    // Some other problem
  }
```

```
} else {
  // Clip found
  val = read_link (sess, ld);
  if (val ["res"] <0) {
    // Error occured
  }
  val ["inf"] ["nf"] - list of material attributes
  val ["inf"] ["lf"] - list of object attributes
  close_link (sess, ld);
}
close_session (sess);
```

Pseudocode for getting a complete list of clips in the database:

```
int sess = open_session ();

array <link_info_t> lst;

// read the root folder
XmlRpcValuie val = list_links (sess, -1, "");
if (val ["res"] == 0) {
  lst + = val ["lst"] ["links"];
}

// gets a list of folders
array <string> folders;
val = list_folders (sess, -1);
if (val ["res"] == 0) {
  folders = val ["lst"] ["folders"];
}

// read folders
for (i = 0; i <folders.size (); i ++) {
  val = list_links (sess, -1, folders [i]);
  if (val ["res"] == 0) {
    lst + = val ["lst"] ["links"];
  }
}

close_session (sess);
```

Examples pseudocode to check for the presence of a clip in the database and read its attributes:

```
int sess = open_session();
XmlRpcValue val = open_link_ns(sess, "title", "Значение заголовка");

int ld = val["res"];
if(ld < 0) {
  if(ld == ERR_NFND) {
    // Not found
  } else {
```

```
    // Some other problem
  }
} else {
  // Clip found
  val = read_link(sess, ld);
  if(val["res"] < 0) {
    // Error occured
  }
  val["inf"]["nf"] - список атрибутов материала
  val["inf"]["lf"] - список атрибутов объекта
  close_link(sess, ld);
}
close_session(sess);
```

Pseudocode to get the full list of clips in the database:

```
int sess = open_session();

array<link_info_t> lst;

//читаем корневую папку
XmlRpcValuie val = list_links(sess, -1, "");
if(val["res"] == 0) {
  lst += val["lst"]["links"];
}

//получает список папок
array<string> folders;
val = list_folders(sess, -1);
if(val["res"] == 0) {
  folders = val["lst"]["folders"];
}

//читаем папки
for(i = 0; i < folders.size(); i++) {
  val = list_links(sess, -1, folders[i]);
  if(val["res"] == 0) {
    lst += val["lst"]["links"];
  }
}

close_session(sess);
```

From:
http://wiki.skylark.tv/ - **wiki.skylark.tv**

Permanent link:
http://wiki.skylark.tv/api/stored

Last update: **2021/12/07 12:33**