

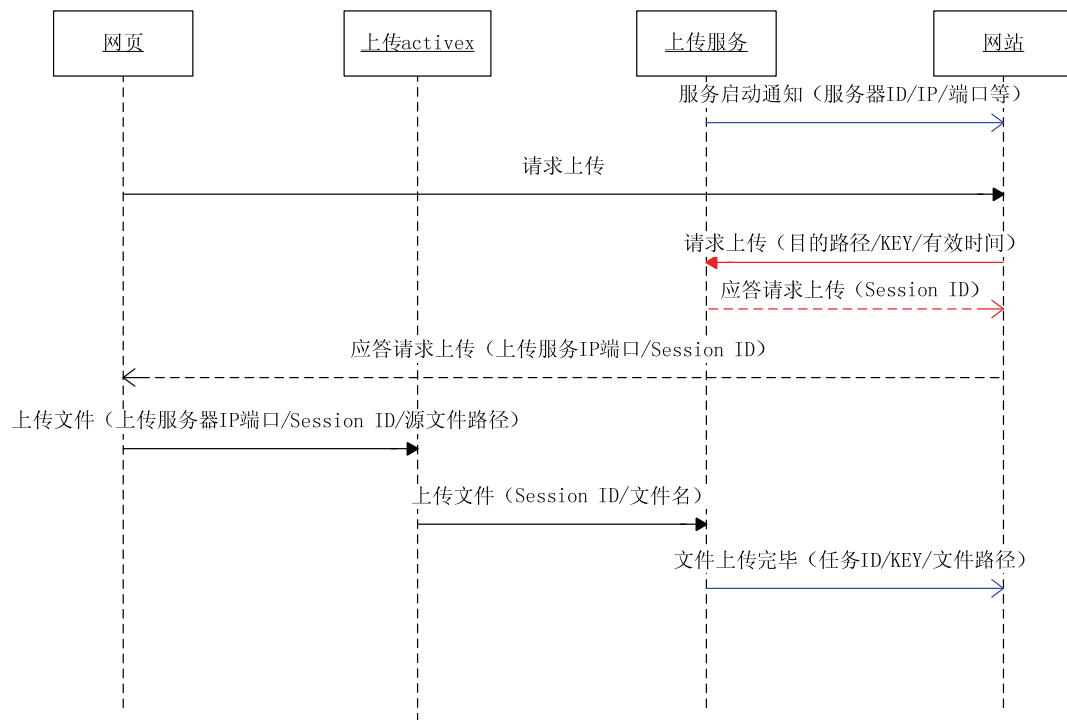
VJVAS SDK

VJVAS is a VOD application server, provide file uploading, recode, picture snapshot and recording features, and provide webservice interface.

Work flow

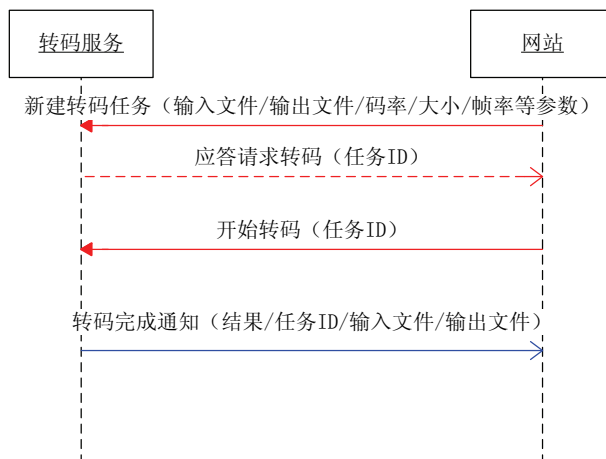
Upload module work flow:

(red line is webservice call, blue line is http report)



Recode module work flow:

(red line is webservice call, blue line is http report)



Webservice Interface

Webservice description address :

<http://nagasoft.cn/download/ws/vjvas3/VJVAS.wsdl>

create upload session

```

int createUploadSession(
char *pwd,           //access password
char *dst_dir,       //upload root directory.
char *key,           //user data, will be used in the http report.
int timeout,         //seconds, =0 no timeout.
int &result          //>=0 return session id, <0 error(-1/-2/-3/-4)
)
    
```

delete upload session

```

int deleteUploadSession(
char *pwd,           //access password
int sid,             //session id
int &result          //>=0 return session id, <0 error (-1/-2/-3)
)
    
```

get upload session

```

struct ns__upload_session{
    int sid;           //session id
    char* dst_dir;     //session root dir
    char* key;         //user key
    int timeout;       //session timeout
};
    
```

```
struct getUploadSessionResponse{
    struct ns__upload_session us;
    int result;          //<0 error(-1/-2/-3)
};
```

```
int getUploadSession(
char *pwd,              //access password
int sid,                //session id
struct ns__getUploadSessionResponse &out
)
```

get upload session list

```
struct upload_session_list{
    struct ns__upload_session* __ptr;
    int __size;
};
```

```
struct getUploadSessionListResponse{
    struct upload_session_list usl;
    int result;          // <0 error (-1/-2/-3)
};
```

```
int ns__getUploadSessionList(
char *pwd,              //access password
struct getUploadSessionListResponse& out //session list
);
```

get upload files list

```
struct ns__upload_session_file{
    char* path;          //file relative path
    int progress;        //upload percent, 0~100
};
```

```
struct upload_session_file_list{
    struct ns__upload_session_file* __ptr;
    int __size;
};
```

```
struct getUploadSessionFilesResponse{
    struct upload_session_file_list usfl;
    int result;          //<0 error (-1/-2/-3)
};
```

```
int getUploadSessionFiles(  
char* pwd,          //access password  
int sid,            //session id  
struct getUploadSessionFilesResponse& out    //file list  
);
```

get system file list

```
struct ns__system_file{  
    bool bDir;          //  
    unsigned long long size; //file size, in byte  
    char* time;          //last modify time, format: CCYY-MM-DD HH:MM:SS  
    char* name;          //file name  
};
```

```
struct system_file_list{  
    struct ns__system_file* __ptr;  
    int __size;  
};
```

```
struct getSystemFilesListResponse{  
    struct system_file_list sfl;  
    int result;          //<0 error (-1/-2/-3/-5)  
};
```

```
int getSystemFilesList(  
char* pwd,          //access password  
char* dir_path,     //absolute path  
struct getSystemFilesListResponse& out //file list  
);
```

delete system file

```
int deleteSystemFile(  
char* pwd,          //access password  
char* path,         //absolute path  
int& result         //<0 error (-1/-2/-3/-5)  
);
```

get system disk driver info

```
struct ns__system_disk_info{  
    char* name;          //driver name, eg:C:\
```

```
    unsigned long long total_size;    //in byte
    unsigned long long free_size;     //in byte
};

struct getSystemDiskInfoResponse{
    struct ns__system_disk_info sdi;
    int result;                       //<0 error (-1/-2/-3)
};

int ns__getSystemDiskInfo(
    char* pwd,                        //access password
    char* path,                       //driver name, eg: C:\
    struct getSystemDiskInfoResponse& out    //result
);
```

get system disk driver list

```
struct system_disk_list{
    struct ns__system_disk_info* __ptr;
    int __size;
};

struct getSystemDiskListResponse{
    struct system_disk_list sdl;
    int result;                       //<0 error (-1/-2/-3)
};

int ns__getSystemDiskList(
    char* pwd,                        //access password
    struct getSystemDiskListResponse& out    //result
);
```

get media file info

```
struct ns__media_file_info{
    int duration;                     //sec
    int width;
    int height;
    char* video_codec;
    int bitrate;                      //kbps
    char* audio_codec;
    float video_framerate;
    int audio_samplerate;
    int audio_bitrate;               //kbps
};
```

```
struct getMediaFileInfoResponse{
    struct ns__media_file_info mfi;
    int result;           //<0 error (-1/-2/-3/-5)
};

int getMediaFileInfo(
char* pwd,               //access password
char* file_path,         //absolute path
struct getMediaFileInfoResponse& out    //result
);
```

picture snapshot (single)

```
struct ns_snap_media_file_pic_in{
    char* path;           //media file absolute path
    char* position;        //snap position, seconds or hh:mm:ss[.xxx], empty for random
    char* size;           // WxH, "0x0" is keep original.
    char* pic_path;        //picture save path, server will select one if set to empty.
    bool bReturnPicData;   //return picture data.
};

struct xsd__base64Binary
{
    unsigned char *__ptr;
    int __size;
};

struct snapMediaFilePictureResponse{
    struct xsd__base64Binary pic;    //picture data.
    int result;                      //<0 error (-1/-2/-3/-5)
};

int snapMediaFilePicture(
char* pwd,                         //access password
struct ns_snap_media_file_pic_in* in, //input parameter
struct snapMediaFilePictureResponse& out    //output parameter
);
```

picture snapshot(multiple)

```
struct ns_snap_media_file_pic_in{
```

```
char* path;          //media file absolute path
char* position;       //snap position, seconds or hh:mm:ss[.xxx], empty for random
char* size;           // WxH, "0x0" is keep original.
char* pic_path;       //picture save path, server will select one if set to empty.
bool bReturnPicData;  //return picture data.
};

struct ns_snap_media_file_pics_in{
    char* path;        //media file absolute path
    char* position;     //snap position, seconds or hh:mm:ss[.xxx], empty for random
    char* size;         // WxH, "0x0" is keep original.
    int count;          //picture counts
    int interval;       //picture interval(ms), 0 for random.
    char* pic_path;     //picture save path, should have wildcards suchas %03d.jpeg. server
    will select one if set to empty.
    bool bReturnPicData; // return picture data.
};

//picture data offset array.
struct pics_offset_in_data{
    int *__ptr;
    int __size;
};

struct snapMediaFilePicturesResponse{
    struct pics_offset_in_data offsets; //the offset in data for every picture.
    struct xsd__base64Binary pics;     //picture data.
    int result;                        //<0 error (-1/-2/-3/-5)
};

//gsoap ns service method-documentation: snap multiple pictures of media file
int snapMediaFilePictures(
    char* pwd,                      //access password
    struct ns_snap_media_file_pics_in* in, //input parameter
    struct snapMediaFilePicturesResponse& out //output parameter
);
```

create recode task

```
struct ns__encode_task_params{
    char* input_file;          //input media file absolute path
    char* output_file;         //output media file absolute path, eg: c:\test.mp4
};
```

```
char* video_codec;      //video codec, eg: h264
int video_bitrate;      //video bitrate, kbps
float frame_rate;       //video framerate, 0.00 for default
char* video_size;       //video size, WxH, 0x0 for default
char* audio_codec;      //audio codec, eg: aac
int audio_bitrate;      //audio bitrate, kbps
int audio_samplerate;    //audio samplerate, 8000/11025/22050/44100/48000...
};
```

please refer to the Appendix for the support input and output files, and video and audio codec list.

```
struct createEncodeTaskResponse{
    int nTaskId;          //task id
    int result;           //<0 error (-1/-2/-3)
};

int createEncodeTask(
char* pwd,               //access password
struct ns__encode_task_params* in, //input parameter
struct createEncodeTaskResponse& out //output parameter
);
```

start recode

```
int startEncodeTask(
char* pwd,               //access password
int nTaskId,             //task ID
int& result               //<0 error (-1/-2/-3)
);
```

stop recode

```
int stopEncodeTask(
char* pwd,               //access password
int nTaskId,             //task ID
int& result               //<0 error (-1/-2/-3)
);
```

delete recode task

```
int deleteEncodeTask(
char* pwd,               //access password
int nTaskId,             //task ID
int& result               //<0 error (-1/-2/-3)
);
```



```
);
```

get recode task

```
struct ns__encode_task{
    int nTaskId;           //task ID
    int state;             //task state, 0=stopped, 1=started, 2=completed.
    struct ns__encode_task_params etp; //record parameters.
};
```

```
struct getEncodeTaskResponse{
    struct ns__encode_task et;
    int result;           //<0 error (-1/-2/-3/-6)
};
```

```
int getEncodeTask(
    char* pwd,           //access password
    int nTaskId,         //task ID
    struct getEncodeTaskResponse& out //output parameter.
);
```

get recode task list

```
struct encode_task_list{
    struct ns__encode_task* __ptr;
    int __size;
};
```

```
struct getEncodeTaskListResponse{
    struct encode_task_list etl;
    int result;           //<0 error (-1/-2/-3)
};
```

```
int getEncodeTaskList(
    char* pwd,           //access password
    struct getEncodeTaskListResponse& out //output parameter.
);
```

get recording channel list

```
struct ns__record_channel{
    int id;               //channel ID, server generated, unique.
    char* name;           //channel name
    char* source_url;     //record url, only support mms protocol now.
};
```

```
char* create_time;           //created time, eg:2010-09-01 11:00:00
};

struct record_channel_list{
    struct ns__record_channel* __ptr;
    int __size;
};

struct getRecordChannelListResponse{
    struct record_channel_list rcl;           //channel list
    int result;                               // <0 error (-1/-2)
};

int getRecordChannelList(
    char* pwd,                               //access password
    struct getRecordChannelListResponse& out //output parameter
);
```

Remarks:

Server will create a internal channel which id is 0, the task created under this channel can be set the url, other channels must use the channels's url.

create recording channel

```
struct create_record_channel_params{
    char* name;           //name
    char* source_url;     //source url, only support mms protocol now.
};

struct createRecordChannelResponse{
    struct ns__record_channel rc;           //output parameter, including the ID.
    int result;                             //<0 error (-1/-2/-3)
};

int createRecordChannel(
    char* pwd,                               //access password
    struct create_record_channel_params* in, //input parameter
    struct createRecordChannelResponse& out //output parameter
);
```

delete recording channel

```
struct deleteRecordChannelResponse{
    int result;           //<0 error (-1/-2/-7)
};
```

```
int deleteRecordChannel(  
    char* pwd,                //access password  
    int channel_id,           //channel id  
    struct deleteRecordChannelResponse& out  
);
```

set recording channel property

```
struct set_record_channel_params{  
    int channel_id;           //channel id  
    char* name;               //name  
    char* source_url;         //source url.  
};  
  
struct setRecordChannelPropertyResponse{  
    struct ns__record_channel rc;    //  
    int result;                     // <0 error (-1/-2/-3/-7)  
};
```

```
int setRecordChannelProperty(  
    char* pwd,                //access password  
    struct set_record_channel_params* in,    //input parameter  
    struct setRecordChannelPropertyResponse& out    //output parameter  
);
```

get recording task list

```
struct record_task_property{  
    char* name;  
    bool auto_record; //  
    char* save_path; //save dir, absolute path or relative path eg:/media/123.wmv.  
                    // Path can contain three wildcards (indicated by% s), the server will  
                    //automatically replace the order: the current date, task name, the  
                    //current time.  
                    //eg: /%s/%s-%s.wmv will replaced as /2010-12-13/name-120000.wmv  
    char* start_time; //start time, eg:2010-09-01 11:00:00.  
                    //ignore for manual task, date ignore for week auto task.  
    char* stop_time; //stop time, eg:2010-09-01 11:00:00.  
                    //ignore for manual task, date ignore for week auto task.  
    int week_flag; // Monday to Sunday for the first 0 to 7, the corresponding bit is set to 1  
                    //ignore for manual task.  
    char* source_url; //source url, ignore if id>0.  
};
```

```
struct ns__record_task{  
    int id;                //task ID
```

```
int state;           //state: 0 stopped, 1 recording
struct record_task_property rtp;    //task property
};

struct record_task_list{
    struct ns__record_task* __ptr;
    int __size;
};

struct getRecordTaskListResponse{
    struct record_task_list rtl;    //task list
    int result;                     //<0 error (-1/-2/-7)
};

int getRecordTaskList(
    char* pwd,                     //access password
    int channel_id,                 //channel id
    struct getRecordTaskListResponse& out    //output parameter.
);

create recording task
struct create_record_task_params{
    int channel_id;                 //channel id
    struct record_task_property rtp; //task property
};

struct createRecordTaskResponse{
    struct ns__record_task rt;      //task info
    int result;                     //<0 error (-1/-2/-3/-7)
};

int createRecordTask(
    char* pwd,                     //access password
    struct create_record_task_params* in, //input parameter
    struct createRecordTaskResponse& out //output parameter
);

modify recording task
struct modify_record_task_params{
    int channel_id;                 //channel ID
    int task_id;                    //task ID
    struct record_task_property rtp; //task proptery, let it empty if you wan't set that
    property
};
```

```
struct modifyRecordTaskResponse{
    struct ns__record_task rt;           //output parameter
    int result;                          //<0 error (-1/-2/-3/-6/-7)
};
```

```
int modifyRecordTask(
    char* pwd,                          //access password
    struct modify_record_task_params* in, //input parameter
    struct modifyRecordTaskResponse& out //output parameter
);
```

delete recoding task

```
struct deleteRecordTaskResponse{
    int result;                          //<0 error (-1/-2/-6/-7)
};
```

```
int deleteRecordTask(
    char* pwd,                          //access password
    int channel_id,                     //channel ID
    int task_id,                        //task ID
    struct deleteRecordTaskResponse& out //output parameter
);
```

start manual task

```
struct startManualRecordTaskResponse{
    int result;                          //<0 error (-1/-2/-6/-8/-9)
};
```

```
Int startManualRecordTask(
    char* pwd,                          //access password
    int channel_id,                     //channel ID
    int task_id,                        //task ID
    struct startManualRecordTaskResponse& out //output parameter
);
```

stop manual task

```
struct stopManualRecordTaskResponse{
    int result;                          //<0 error (-1/-2/-6/-8)
};
```

```
int stopManualRecordTask(
    char* pwd,                          //access password
    int channel_id,                     //channel ID
```

```
int task_id, //task ID
struct stopManualRecordTaskResponse& out //output parameter
);
```

get server information

```
struct license_info{
    char* user_id; //User ID
    char* user_name; //User Name
    char* install_date; //License installed time
    int days; //can used days
    char* host_id; //Host Id
    char* desc; //Description
};
```

```
struct vas_server_info{
    int server_id; //Server ID
    int upload_port; //upload port
    char* record_root_dir; //recording root dir
    char* media_root_dir; //media root dir
    struct license_info li; //
};
```

```
struct getVasServerInfoResponse{
    struct vas_server_info si; //output parameter
    int result; //<0 error (-1)
};
```

```
int getVasServerInfo(
    char* pwd, //access password
    struct getVasServerInfoResponse& out //output parameter
);
```

Return values:

```
#define WS_SUCCESS 0
#define WS_ERROR_INVALID_PWD -1
#define WS_ERROR_UNKOWN -2
#define WS_ERROR_INVALID_PARAM -3
#define WS_ERROR_CREATE_DIR_FAILED -4
#define WS_ERROR_PATH_NOT_EXIST -5
#define WS_ERROR_TASK_NOT_EXIT -6
#define WS_ERROR_CHANNEL_NOT_EXIST -7
#define WS_ERROR_NOT_MANUAL_TASK -8
#define WS_ERROR_TASK_START_FAILED -9
```

HTTP Report

HTTP Report is used to notify web there is event happened, use HTTP POST method, UTF-8 encoding. The http report address can set in the server configuration file.

1. Server startup

app=vjvas&msg=1&serverid=?&wsip=?&wsport=?&wspwd=?&uploadip=?&uploadport=?

serverid Server ID

wsip webservice listen ip.

wsport webservice listen port.

wspwd webservice password

uploadipupload listen ip

uploadport upload listen port

2. Server shutdown

app=vjvas&msg=2&serverid=?

serverid server id

3. New upload file

app=vjvas&msg=3&serverid=?&sid=?&key=?&path=?

sid session id

key user key

path relative path

4. File upload completed

app=vjvas&msg=4&serverid=?&sid=?&key=?&path=?

sid session id

key user key

path relative path

5. Create directory

app=vjvas&msg=5&serverid=?&sid=?&key=?&path=?&isdir=1

sid session id

key user key

path relative path

6. Rename file

app=vjvas&msg=6&serverid=?&sid=?&key=?&srcpath=?&dstname=?&isdir=1|0

sid session id

key user key

srcpath old name, relative path

dstname new name, relative path

7. Delete file

app=vjvas&msg=7&serverid=?&sid=?&key=?&path=?&isdir=1|0

sid session id

key user key

path relative path

8. Move file

app=vjvas&msg=8&serverid=?&sid=?&key=?&srcpath=?&dstpath=?&isdir=1|0

sid session id

key user key

srcpath old relative path

dstname new relative path

9. Record completed

app=vjvas&msg=9&serverid=?&taskid=?&inputfile=?&outputfile=?&result=?

taskid task id

inputfile input file

outputfile output file

result =0 success !=0 failed.

10. Recording start

app=vjvas&msg=10&serverid=?&channelid=?&taskid=?&channelname=?&taskname=?&begintime=?&outputfile=?&result=?

channelid channel ID

taskid Task ID

channelname channel name

taskname task name

begintime start time, format: 2010-12-13 12:00:00

outputfile output path, absolute path.

result =0 success, !=0 failed

11. Recording stop

app=vjvas&msg=11&serverid=?&channelid=?&taskid=?&channelname=?&taskname=?&begintime=?&endtime=?&outputfile=?&result=?

channelid channel ID

taskid Task ID

channelname channel name

taskname task name

begintime start time, format: 2010-12-13 12:00:00

endtime stop time, format: 2010-12-13 13:00:00

outputfile output path, absolute path.

result =0 success, !=0 failed

upload activex

Name:"VJUploadAx Class"

Progid:"upax.VJUploadAx.1"

Clsid:"365B9C56-F7BE-4789-8C8B-36E8A9453087"

Property:

ID	Name	Description	Remark
1	Server	Upload sever address, "ip:port"	Eg:192.168.0.1:5030
2	SessionID	Upload session id	
3	EnableSelectDirectory		
4	EnableUseMD5ForName	Use md5 for file name	If set to true, will not display the server file list
5	EnableCreateDirectory		Can't upload directory if set to false.
6	EnableRenameFile		
7	EnableMoveFile		
8	EnableDeleteFile		
9	FileFilterString	Filter string when upload directory, Format: *.wmv;*.rm;...*.avi;	

Example

1.snap picture

```

struct ns_snap_media_file_pic_in smfpi;
smfpi.path = "f:\\media\\3.rmvb";
smfpi.position = "";/"00:10:00.000";
smfpi.size = "";
smfpi.pic_path = "";/"c:\\temp\\test.jpeg";
smfpi.bReturnPicData = true;
struct snapMediaFilePictureResponse mfpr;
n = sp.snapMediaFilePicture(pwd, &smfpi, mfpr);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
else if (mfpr.result < 0){
    std::cout << "snapMediaFilePicture error : " << mfpr.result << std::endl;
}
else{
    std::cout << "snapMediaFilePicture success" << std::endl;

    FILE *pf = fopen("c:\\temp\\snap-out.jpeg", "w+b");
    if (pf){
        fwrite(mfpr.pic.__ptr, 1, mfpr.pic.__size, pf);
    }
}

```

```
        fclose(pf);
    }
}

struct ns_snap_media_file_pics_in smfpsi;
smfpsi.path = "f:\\media\\3.rmvb";
smfpsi.position = ""; // "00:10:00.000";
smfpsi.count = 5;
smfpsi.interval = 2000;
smfpsi.size = "";
smfpsi.pic_path = ""; // "c:\\temp\\test-%03d.jpeg";
smfpsi.bReturnPicData = true;
struct snapMediaFilePicturesResponse mfpsr;
n = sp.snapMediaFilePictures(pwd, &smfpsi, mfpsr);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
else if (mfpsr.result < 0){
    std::cout << "snapMediaFilePicture error : " << mfpsr.result << std::endl;
}
else{
    std::cout << "snapMediaFilePicture success" << std::endl;
    for (int i=0; i<mfpsr.offsets.__size; ++i){
        char pszname[100];
        sprintf(pszname, "c:\\temp\\snap-out-%03d.jpeg", i);
        FILE *pf = fopen(pszname, "w+b");
        if (pf){
            int nNextOffset = (i < mfpsr.offsets.__size-1) ? mfpsr.offsets.__ptr[i+1] :
mfpsr.pics.__size;
            int nPicSize = nNextOffset - mfpsr.offsets.__ptr[i];
            fwrite(mfpsr.pics.__ptr+mfpsr.offsets.__ptr[i], 1, nPicSize, pf);
            fclose(pf);
        }
    }
}
```

2.Recode

```
struct ns__encode_task_params etp;
etp.input_file = "f:\\media\\dvavi\\blue.avi";
etp.output_file = "c:\\temp\\test-output.mp4";
etp.video_codec = "h264";
etp.video_bitrate = 400;
etp.frame_rate = 25.0f;
etp.video_size = "320x240";
etp.audio_codec = "aac";
```

```
etp.audio_bitrate = 32;
etp.audio_samplerate = 22050;
struct createEncodeTaskResponse etr;
n = sp.createEncodeTask(pwd, &etp, etr);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
else if (etr.result < 0){
    std::cout << "createEncodeTask error : " << etr.result << std::endl;
}
else{
    std::cout << "createEncodeTask success" << std::endl;
    std::cout << "task id: " << etr.nTaskId << std::endl;
}

n = sp.startEncodeTask(pwd, etr.nTaskId, result);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
else if (result < 0){
    std::cout << "startEncodeTask error : " << result << std::endl;
}
else{
    std::cout << "startEncodeTask success" << std::endl;
}

struct getEncodeTaskResponse getr;
n = sp.getEncodeTask(pwd, etr.nTaskId, getr);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
else if (getr.result < 0){
    std::cout << "getEncodeTask error : " << getr.result << std::endl;
}
else{
    std::cout << "getEncodeTask success" << std::endl;
    std::cout << "task id: " << getr.et.nTaskId << std::endl;
    std::cout << "state: " << getr.et.state << std::endl;
    std::cout << "params: " << std::endl;
    std::cout << "input_file: " << getr.et.etp.input_file << std::endl;
    std::cout << "output_file: " << getr.et.etp.output_file << std::endl;
    std::cout << "video_codec: " << getr.et.etp.video_codec << std::endl;
    std::cout << "video_bitrate: " << getr.et.etp.video_bitrate << std::endl;
    std::cout << "framerate: " << getr.et.etp.frame_rate << std::endl;
}
```

```
std::cout << "size: " << getr.etp.video_size << std::endl;
std::cout << "audio_codec: " << getr.etp.audio_codec << std::endl;
std::cout << "audio_bitrate: " << getr.etp.audio_bitrate << std::endl;
std::cout << "audio_samplerate: " << getr.etp.audio_samplerate << std::endl;
}

struct getEncodeTaskListResponse getlr;
n = sp.getEncodeTaskList(pwd, getlr);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
else if (getlr.result < 0){
    std::cout << "getEncodeTaskList error : " << getlr.result << std::endl;
}
else{
    std::cout << "getEncodeTaskList success" << std::endl;

    for (int i=0; i < getlr.etl.__size; ++i){
        std::cout << std::endl;
        std::cout << "task id: " << getlr.etl.__ptr[0].nTaskId << std::endl;
        std::cout << "state: " << getlr.etl.__ptr[0].state << std::endl;
        std::cout << "params: " << std::endl;
        std::cout << "input_file: " << getlr.etl.__ptr[0].etp.input_file << std::endl;
        std::cout << "output_file: " << getlr.etl.__ptr[0].etp.output_file <<
std::endl;
        std::cout << "video_codec: " << getlr.etl.__ptr[0].etp.video_codec <<
std::endl;
        std::cout << "video_bitrate: " << getlr.etl.__ptr[0].etp.video_bitrate <<
std::endl;
        std::cout << "framerate: " << getlr.etl.__ptr[0].etp.frame_rate << std::endl;
        std::cout << "size: " << getlr.etl.__ptr[0].etp.video_size << std::endl;
        std::cout << "audio_codec: " << getlr.etl.__ptr[0].etp.audio_codec <<
std::endl;
        std::cout << "audio_bitrate: " << getlr.etl.__ptr[0].etp.audio_bitrate <<
std::endl;
        std::cout << "audio_samplerate: " <<
getlr.etl.__ptr[0].etp.audio_samplerate << std::endl;
    }
}

n = sp.stopEncodeTask(pwd, etr.nTaskId, result);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
```

```
else if (result < 0){
    std::cout << "stopEncodeTask error : " << result << std::endl;
}
else{
    std::cout << "stopEncodeTask success" << std::endl;
}

n = sp.deleteEncodeTask(pwd, etr.nTaskId, result);
if (n != 0 && sp.error){
    sp.soap_stream_fault(std::cerr);
}
else if (result < 0){
    std::cout << "deleteEncodeTask error : " << result << std::endl;
}
else{
    std::cout << "deleteEncodeTask success" << std::endl;
}
```

Appendix

1.supported input and output files(E output, D input)。

E 3g2	3GP2 format
E 3gp	3GP format
D 4xm	4X Technologies format
D IFF	IFF format
D ISS	Funcom ISS format
D MTV	MTV format
DE RoQ	raw id RoQ format
D aac	raw ADTS AAC
DE ac3	raw AC-3
E adts	ADTS AAC
D aea	MD STUDIO audio
DE aiff	Audio IFF
DE alaw	PCM A-law format
DE amr	3GPP AMR file format
D anm	Deluxe Paint Animation
D apc	CRYO APC format
D ape	Monkey's Audio
DE asf	ASF format
E asf_stream	ASF format
DE ass	SSA/ASS format
DE au	SUN AU format
DE avi	AVI format
E avm2	Flash 9 (AVM2) format
D avs	AVISynth

D bethsoftvid	Bethesda Softworks VID format
D bfi	Brute Force & Ignorance
D bink	Bink
D c93	Interplay C93
D caf	Apple Core Audio Format
D cavsvideo	raw Chinese AVS video
D cdg	CD Graphics Format
E crc	CRC testing format
DE daud	D-Cinema audio format
DE dirac	raw Dirac
DE dnxhd	raw DNxHD (SMPTE VC-3)
D dsicin	Delphine Software International CIN format
DE dts	raw DTS
DE dv	DV video format
E dvd	MPEG-2 PS format (DVD VOB)
D dxa	DXA
D ea	Electronic Arts Multimedia Format
D ea_cdata	Electronic Arts cdata
DE eac3	raw E-AC-3
DE f32be	PCM 32 bit floating-point big-endian format
DE f32le	PCM 32 bit floating-point little-endian format
DE f64be	PCM 64 bit floating-point big-endian format
DE f64le	PCM 64 bit floating-point little-endian format
DE ffm	FFM (FFserver live feed) format
D film_cpk	Sega FILM/CPK format
DE filmstrip	Adobe Filmstrip
DE flac	raw FLAC
D flic	FLI/FLC/FLX animation format
DE flv	FLV format
E framecrc	framecrc testing format
E gif	GIF Animation
D gsm	raw GSM
DE gxf	GXF format
DE h261	raw H.261
DE h263	raw H.263
DE h264	raw H.264 video format
D idcin	id Cinematic format
DE image2	image2 sequence
DE image2pipe	piped image2 sequence
D ingenient	raw Ingenient MJPEG
D ipmovie	Interplay MVE format
E ipod	iPod H.264 MP4 format
D iv8	A format generated by IndigoVision 8000 video server
D lmlm4	lmlm4 raw format

DE m4v	raw MPEG-4 video format
DE matroska	Matroska file format
E md5	MD5 testing format
DE mjpeg	raw MJPEG video
DE mlp	raw MLP
D mm	American Laser Games MM format
DE mmf	Yamaha SMAF
E mov	MOV format
D mov,mp4,m4a,3gp,3g2,mj2	QuickTime/MPEG-4/Motion JPEG 2000 format
E mp2	MPEG audio layer 2
DE mp3	MPEG audio layer 3
E mp4	MP4 format
D mpc	Musepack
D mpc8	Musepack SV8
DE mpeg	MPEG-1 System format
E mpeg1video	raw MPEG-1 video
E mpeg2video	raw MPEG-2 video
DE mpegts	MPEG-2 transport stream format
D mpegtsraw	MPEG-2 raw transport stream format
D mpegvideo	raw MPEG video
E mpjpeg	MIME multipart JPEG format
D msnwctcp	MSN TCP Webcam stream
DE mulaw	PCM mu-law format
D mvi	Motion Pixels MVI format
DE mxf	Material eXchange Format
E mxf_d10	Material eXchange Format, D-10 Mapping
D nc	NC camera feed format
D nsv	Nullsoft Streaming Video
E null	raw null video format
DE nut	NUT format
D nuv	NuppelVideo format
DE ogg	Ogg
D oma	Sony OpenMG audio
E psp	PSP MP4 format
D psxstr	Sony Playstation STR format
D pva	TechnoTrend PVA file and stream format
D qcp	QCP format
D r3d	REDCODE R3D format
DE rawvideo	raw video format
E rcv	VC-1 test bitstream
D rl2	RL2 format
DE rm	RealMedia format
D rpl	RPL/ARMovie format
E rtp	RTP output format

DE rtsp	RTSP output format
DE s16be	PCM signed 16 bit big-endian format
DE s16le	PCM signed 16 bit little-endian format
DE s24be	PCM signed 24 bit big-endian format
DE s24le	PCM signed 24 bit little-endian format
DE s32be	PCM signed 32 bit big-endian format
DE s32le	PCM signed 32 bit little-endian format
DE s8	PCM signed 8 bit format
D sdp	SDP
D shn	raw Shorten
D siff	Beam Software SIFF
D smk	Smacker video
D sol	Sierra SOL format
DE sox	SoX native format
E spdif	IEC958 - S/PDIF (IEC-61937)
E svcd	MPEG-2 PS format (VOB)
DE swf	Flash format
D thp	THP
D tiertexseq	Tiertex Limited SEQ format
D tmv	8088flex TMV
DE truehd	raw TrueHD
D tta	True Audio
D txd	Renderware TeXture Dictionary
DE u16be	PCM unsigned 16 bit big-endian format
DE u16le	PCM unsigned 16 bit little-endian format
DE u24be	PCM unsigned 24 bit big-endian format
DE u24le	PCM unsigned 24 bit little-endian format
DE u32be	PCM unsigned 32 bit big-endian format
DE u32le	PCM unsigned 32 bit little-endian format
DE u8	PCM unsigned 8 bit format
D vc1	raw VC-1
D vc1test	VC-1 test bitstream format
E vcd	MPEG-1 System format (VCD)
D vfwcap	VFW video capture
D vmd	Sierra VMD format
E vob	MPEG-2 PS format (VOB)
DE voc	Creative Voice file format
D vqf	Nippon Telegraph and Telephone Corporation (NTT)
TwinVQ	
D w64	Sony Wave64 format
DE wav	WAV format
D wc3movie	Wing Commander III movie format
D wsaud	Westwood Studios audio format
D wsvqa	Westwood Studios VQA format

D	wv	WavPack
D	xa	Maxis XA File Format
D	yop	Psygnosis YOP Format
DE	yuv4mpegpipe	YUV4MPEG pipe format

2.support video and audio codec

D decoder

E Encoder

V video

A audio

D V D	4xm	4X Movie
D V D	8bps	QuickTime 8BPS video
D A	8svx_exp	8SVX exponential
D A	8svx_fib	8SVX fibonacci
D V D	FRWU	Forward Uncompressed
DEA	aac	Advanced Audio Coding
D V D	aasc	Autodesk RLE
DEA	ac3	ATSC A/52A (AC-3)
D A	adpcm_4xm	ADPCM 4X Movie
DEA	adpcm_adx	SEGA CRI ADX ADPCM
D A	adpcm_ct	ADPCM Creative Technology
D A	adpcm_ea	ADPCM Electronic Arts
D A	adpcm_ea_maxis_xa	ADPCM Electronic Arts Maxis CDROM XA
D A	adpcm_ea_r1	ADPCM Electronic Arts R1
D A	adpcm_ea_r2	ADPCM Electronic Arts R2
D A	adpcm_ea_r3	ADPCM Electronic Arts R3
D A	adpcm_ea_xas	ADPCM Electronic Arts XAS
D A	adpcm_ima_amv	ADPCM IMA AMV
D A	adpcm_ima_dk3	ADPCM IMA Duck DK3
D A	adpcm_ima_dk4	ADPCM IMA Duck DK4
D A	adpcm_ima_ea_eacs	ADPCM IMA Electronic Arts EACS
D A	adpcm_ima_ea_sead	ADPCM IMA Electronic Arts SEAD
D A	adpcm_ima_iss	ADPCM IMA Funcom ISS
DEA	adpcm_ima_qt	ADPCM IMA QuickTime
D A	adpcm_ima_smjpeg	ADPCM IMA Loki SDL MJPEG
DEA	adpcm_ima_wav	ADPCM IMA WAV
D A	adpcm_ima_ws	ADPCM IMA Westwood
DEA	adpcm_ms	ADPCM Microsoft
D A	adpcm_sbpro_2	ADPCM Sound Blaster Pro 2-bit
D A	adpcm_sbpro_3	ADPCM Sound Blaster Pro 2.6-bit
D A	adpcm_sbpro_4	ADPCM Sound Blaster Pro 4-bit
DEA	adpcm_swf	ADPCM Shockwave Flash
D A	adpcm_thp	ADPCM Nintendo Gamecube THP

D A	adpcm_xa	ADPCM CDROM XA
DEA	adpcm_yamaha	ADPCM Yamaha
DEA	alac	ALAC (Apple Lossless Audio Codec)
D A	als	MPEG-4 Audio Lossless Coding (ALS)
D A	amrnb	Adaptive Multi-Rate NarrowBand
D V D	amv	AMV Video
D V D	anm	Deluxe Paint Animation
D A	ape	Monkey's Audio
DEV D	asv1	ASUS V1
DEV D	asv2	ASUS V2
D A	atrac1	Atrac 1 (Adaptive TRansform Acoustic Coding)
D A	atrac3	Atrac 3 (Adaptive TRansform Acoustic Coding 3)
D V D	aura	Auravision AURA
D V D	aura2	Auravision Aura 2
D V D	avs	AVS (Audio Video Standard) video
D V D	bethsoftvid	Bethesda VID video
D V D	bfi	Brute Force & Ignorance
D A	binkaudio_dct	Bink Audio (DCT)
D A	binkaudio_rdft	Bink Audio (RDFT)
D V	binkvideo	Bink video
DEV D	bmp	BMP image
D V D	c93	Interplay C93
D V D	camstudio	CamStudio
D V D	camtasia	TechSmith Screen Capture Codec
D V D	cavs	Chinese AVS video (AVS1-P2, JiZhun profile)
D V D	cdgraphics	CD Graphics video
D V D	cinepak	Cinepak
D V D	cljr	Cirrus Logic AccuPak
D A	cook	COOK
D V D	cyuv	Creative YUV (CYUV)
D A	dca	DCA (DTS Coherent Acoustics)
DEV D	dnxhd	VC3/DNxHD
D V	dpx	DPX image
D A	dsicinaudio	Delphine Software International CIN audio
D V D	dsicinvideo	Delphine Software International CIN video
DES	dvbsub	DVB subtitles
DES	dvdsup	DVD subtitles
DEV D	dvvideo	DV (Digital Video)
D V D	dxs	Feeble Files/ScummVM DXA
D A	eac3	ATSC A/52B (AC-3, E-AC-3)
D V D	eacmv	Electronic Arts CMV video
D V D	eamad	Electronic Arts Madcow Video
D V D	eatgq	Electronic Arts TGQ video
D V	eatgv	Electronic Arts TGV video

D V D	eatqi	Electronic Arts TQI Video
D V D	escape124	Escape 124
DEV D	ffv1	FFmpeg video codec #1
DEVSD	ffvhuff	Huffyuv FFmpeg variant
DEA	flac	FLAC (Free Lossless Audio Codec)
DEV D	flashsv	Flash Screen Video
D V D	flic	Autodesk Animator Flic video
DEVSD	flv	Flash Video (FLV) / Sorenson Spark / Sorenson H.263
D V D	fraps	Fraps
DEA	g726	G.726 ADPCM
DEV D	gif	GIF (Graphics Interchange Format)
DEV D	h261	H.261
DEVSDT	h263	H.263 / H.263-1996
D VSD	h263i	Intel H.263
EV	h263p	H.263+ / H.263-1998 / H.263 version 2
D V D	h264	H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10
DEVSD	huffyuv	Huffyuv / HuffYUV
D V D	idcinvideo	id Quake II CIN video
D V D	iff_byterun1	IFF ByteRun1
D V D	iff_ilbm	IFF ILBM
D A	imc	IMC (Intel Music Coder)
D V D	indeo2	Intel Indeo 2
D V D	indeo3	Intel Indeo 3
D V	indeo5	Intel Indeo Video Interactive 5
D A	interplay_dpcm	DPCM Interplay
D V D	interplayvideo	Interplay MVE video
DEV D	jpegls	JPEG-LS
D V	kgv1	Kega Game Video
D V D	kmvc	Karl Morton's video codec
D A	libfaad	libfaad AAC (Advanced Audio Codec)
DEA	libgsm	libgsm GSM
DEA	libgsm_ms	libgsm GSM Microsoft variant
EA	libmp3lame	libmp3lame MP3 (MPEG audio layer 3)
DEA	libopencore_amrnb	OpenCORE Adaptive Multi-Rate (AMR)
Narrow-Band		
D A	libopencore_amrwb	OpenCORE Adaptive Multi-Rate (AMR) Wide-Band
D V D	libopenjpeg	OpenJPEG based JPEG 2000 decoder
DEV	libschrödinger	libschrödinger Dirac 2.2
D A	libspeex	libspeex Speex
EV	libtheora	libtheora Theora
EA	libvorbis	libvorbis Vorbis
EV	libx264	libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10
EV	libxvid	libxvidcore MPEG-4 part 2
EV	ljpeg	Lossless JPEG

D V D	loco	LOCO
D A	mace3	MACE (Macintosh Audio Compression/Expansion) 3:1
D A	mace6	MACE (Macintosh Audio Compression/Expansion) 6:1
D V D	mdec	Sony PlayStation MDEC (Motion DECoder)
D V D	mimic	Mimic
DEV D	mjpeg	MJPEG (Motion JPEG)
D V D	mjpegb	Apple MJPEG-B
D A	mlp	MLP (Meridian Lossless Packing)
D V D	mmvideo	American Laser Games MM Video
D V D	motionpixels	Motion Pixels video
D A	mp1	MP1 (MPEG audio layer 1)
D A	mp1float	MP1 (MPEG audio layer 1)
DEA	mp2	MP2 (MPEG audio layer 2)
D A	mp2float	MP2 (MPEG audio layer 2)
D A	mp3	MP3 (MPEG audio layer 3)
D A	mp3adu	ADU (Application Data Unit) MP3 (MPEG audio layer 3)
D A	mp3adufloat	ADU (Application Data Unit) MP3 (MPEG audio layer 3)
D A	mp3float	MP3 (MPEG audio layer 3)
D A	mp3on4	MP3onMP4
D A	mp3on4float	MP3onMP4
D A	mpc7	Musepack SV7
D A	mpc8	Musepack SV8
DEVSDT	mpeg1video	MPEG-1 video
DEVSDT	mpeg2video	MPEG-2 video
DEVSDT	mpeg4	MPEG-4 part 2
D VSDT	mpegvideo	MPEG-1 video
DEVSD	msmpeg4	MPEG-4 part 2 Microsoft variant version 3
DEVSD	msmpeg4v1	MPEG-4 part 2 Microsoft variant version 1
DEVSD	msmpeg4v2	MPEG-4 part 2 Microsoft variant version 2
D V D	msrle	Microsoft RLE
D V D	msvideo1	Microsoft Video 1
D V D	mszh	LCL (LossLess Codec Library) MSZH
DEA	nellymoser	Nellymoser Asao
D V D	nuv	NuppelVideo/RTJPEG
DEV D	pam	PAM (Portable AnyMap) image
DEV D	pbm	PBM (Portable BitMap) image
DEA	pcm_alaw	PCM A-law
D A	pcm_bluray	PCM signed 16 20 24-bit big-endian for Blu-ray
media		
D A	pcm_dvd	PCM signed 20 24-bit big-endian
DEA	pcm_f32be	PCM 32-bit floating point big-endian
DEA	pcm_f32le	PCM 32-bit floating point little-endian

DEA	pcm_f64be	PCM 64-bit floating point big-endian
DEA	pcm_f64le	PCM 64-bit floating point little-endian
DEA	pcm_mulaw	PCM mu-law
DEA	pcm_s16be	PCM signed 16-bit big-endian
DEA	pcm_s16le	PCM signed 16-bit little-endian
D A	pcm_s16le_planar	PCM 16-bit little-endian planar
DEA	pcm_s24be	PCM signed 24-bit big-endian
DEA	pcm_s24daud	PCM D-Cinema audio signed 24-bit
DEA	pcm_s24le	PCM signed 24-bit little-endian
DEA	pcm_s32be	PCM signed 32-bit big-endian
DEA	pcm_s32le	PCM signed 32-bit little-endian
DEA	pcm_s8	PCM signed 8-bit
DEA	pcm_u16be	PCM unsigned 16-bit big-endian
DEA	pcm_u16le	PCM unsigned 16-bit little-endian
DEA	pcm_u24be	PCM unsigned 24-bit big-endian
DEA	pcm_u24le	PCM unsigned 24-bit little-endian
DEA	pcm_u32be	PCM unsigned 32-bit big-endian
DEA	pcm_u32le	PCM unsigned 32-bit little-endian
DEA	pcm_u8	PCM unsigned 8-bit
DEA	pcm_zork	PCM Zork
DEV D	pcx	PC Paintbrush PCX image
DEV D	pgm	PGM (Portable GrayMap) image
DEV D	pgmyuv	PGMYUV (Portable GrayMap YUV) image
D S	pgssub	HDMV Presentation Graphic Stream subtitles
DEV D	png	PNG image
DEV D	ppm	PPM (Portable PixelMap) image
D V D	ptx	V.Flash PTX image
D A	qcelp	QCELP / PureVoice
D A	qdm2	QDesign Music Codec 2
D V D	qdraw	Apple QuickDraw
D V D	qpeg	Q-team QPEG
DEV D	qtrle	QuickTime Animation (RLE) video
D V D	r210	Uncompressed RGB 10-bit
DEV	rawvideo	raw video
D A	real_144	RealAudio 1.0 (14.4K)
D A	real_288	RealAudio 2.0 (28.8K)
D V D	rl2	RL2 video
DEA	roq_dpcm	id RoQ DPCM
DEV D	roqvideo	id RoQ video
D V D	rpza	QuickTime video (RPZA)
DEV D	rv10	RealVideo 1.0
DEV D	rv20	RealVideo 2.0
D V D	rv30	RealVideo 3.0
D V D	rv40	RealVideo 4.0

DEV	sgi	SGI image
D A	shorten	Shorten
D A	sipr	RealAudio SIPR / ACELP.NET
D A	smackaud	Smacker audio
D V D	smackvid	Smacker video
D V D	smc	QuickTime Graphics (SMC)
DEV D	snow	Snow
D A	sol_dpcm	DPCM Sol
DEA	sonic	Sonic
EA	sonicls	Sonic lossless
D V D	sp5x	Sunplus JPEG (SP5X)
D V D	sunrast	Sun Rasterfile image
DEV D	svq1	Sorenson Vector Quantizer 1 / Sorenson Video 1 / SVQ1
D VSD	svq3	Sorenson Vector Quantizer 3 / Sorenson Video 3 / SVQ3
DEV D	targa	Truevision Targa image
D VSD	theora	Theora
D V D	thp	Nintendo Gamecube THP video
D V D	tiertexseqvideo	Tiertex Limited SEQ video
DEV D	tiff	TIFF image
D V D	tmv	8088flex TMV
D A	truehd	TrueHD
D V D	truemotion1	Duck TrueMotion 1.0
D V D	truemotion2	Duck TrueMotion 2.0
D A	truespeech	DSP Group TrueSpeech
D A	tta	True Audio (TTA)
D A	twinvq	VQF TwinVQ
D V D	txd	Renderware TXD (TeXture Dictionary) image
D V D	ultimotion	IBM UltiMotion
DEV D	v210	Uncompressed 4:2:2 10-bit
D V D	v210x	Uncompressed 4:2:2 10-bit
D V	vb	Beam Software VB
D V D	vc1	SMPTE VC-1
D V D	vcr1	ATI VCR1
D A	vmdataudio	Sierra VMD audio
D V D	vmdvideo	Sierra VMD video
D V D	vmnc	VMware Screen Codec / VMware Video
DEA	vorbis	Vorbis
D VSD	vp3	On2 VP3
D V D	vp5	On2 VP5
D V D	vp6	On2 VP6
D V D	vp6a	On2 VP6 (Flash version, with alpha channel)
D V D	vp6f	On2 VP6 (Flash version)

D V D	vqavideo	Westwood Studios VQA (Vector Quantized Animation)
video		
D A	wavpack	WavPack
D A	wmapro	Windows Media Audio 9 Professional
DEA	wmav1	Windows Media Audio 1
DEA	wmav2	Windows Media Audio 2
D A	wmavoice	Windows Media Audio Voice
DEVSD	wmv1	Windows Media Video 7
DEVSD	wmv2	Windows Media Video 8
D V D	wmv3	Windows Media Video 9
D V D	wnv1	Winnov WNV1
D A	ws_snd1	Westwood Audio (SND1)
D A	xan_dpcm	DPCM Xan
D V D	xan_wc3	Wing Commander III / Xan
D V D	xl	Miro VideoXL
DES	xsub	DivX subtitles (XSUB)
D V	yop	Psygnosis YOP Video
DEV D	zlib	LCL (LossLess Codec Library) ZLIB
DEV D	zmbv	Zip Motion Blocks Video