

The programming interface TELE2 API

Introduction

The programming interface TELE2 API is used for:

- Creating own applications, using TELE2 functions (playback of video-clips with overlaid upper-level compositions, different effects)
- Customization of the TELE itself, for example to add accounting or billing system, etc.

The advantage of TELE2 API is that all its functions return control immediately, so your program will not 'hang'.

The application can be written in any programming language that supports integration of ActiveX controls, COM-objects or DLLs. For example, the following programming means can be used: DHTML (VB, JavaScript), VBA (MS Word, MS Excel, MS Access), Microsoft Visual Basic 6.0/2003/2005, Microsoft C++ 6.0/2003/2005, Microsoft C# 6.0/2003/2005, Borland Delphi (for Windows 32/64).

For interface-oriented programming languages such as DHTML, VBA, VB, C# it is more convenient to use ActiveX TELE2Ctrl control. To add it to a program one can add it to the Toolbox panel (in VS2005 it is done by pressing the right button on it and selecting "Choose Items" – then one should choose COM TELE2Ctrl) and drag it to the current dialog of the program.

So, one can call the object in the following ways:

- a) by VBasic CreateObject("TELE.API") commant
- b) by ActiveX TELE2Ctrl control
- c) directly through Logo.dll, which includes both interfaces and can be imported directly as a DLL.

The other advantage of using TELE2Ctrl comparing to the other ways of calling TELE2 API is the support of events. For example, event FastTimer is called exactly when frame begins.

TELE2 API functionality includes:

- Control over schedule playback TELE2 (start from any moment, stop, start of any element alone)
- Loading, saving, schedule creating, creating of clips and upper layer compositions
- Calling different interfaces of TELE2 program

There are 2 modes of TELE2 API usage: with TELE2 running and without it. As it was said, the interface can be used for control/customization of TELE and for creating own applications using TELE2 functions. According to that, some functions can call running TELE2 application (i.e. TELE2 must be running), and the other part of functions doesn't require it, because it works independently.

The functions working independently from TELE2 and not requiring it to be running are:

```
Login  
AssignValue  
GetValue  
LoadLogosAndProfiles  
EditLogotype  
SendObjDirect  
SendObjects  
SendPlayWithLogo  
FileClose  
FileWriteLine  
GetFilePath
```

Functions for control over TELE2 that require TELE2 to be running:

```
AProAssignValue  
AProGetValue  
AProExecCommand  
AProGetStatus  
AddSchedItem  
DelSchedItem  
EditBlock  
EditClip  
FindSchedItem  
PlayBlock  
PlayClip  
SetItemLogo  
SetItemProfile  
StartPlaybackFromItem  
StartPlaybackFromTime  
StartSystemPlay  
StopPlayback
```

Any program must call Login function in the very beginning, to get access to the TELE2API. This function is called in both cases – if it uses TELE2 program or if it is working independently and uses only TELE2 API. The upper level compositions are stored in XML files. TELE itself stores them in LogosAndProfiles.XML file, which is placed in the Alphapro directory. Nevertheless, programs that work independently from TELE can use any other file for that. All that should be done in this case is to load it by the command LoadLogosAndProfiles. One can load and use LogosAndProfiles.XML, that is used by TELE program, but do not forget that program works with data loaded to RAM, so if your application will load LogosAndProfiles.XML, there would be 2 copies of that file in memory – one loaded by TELE, the other – by your application; and changing of same objects or logotypes by one program will not cause automatic change of them by the other program – it will be necessary to save the changed LogosAndProfiles.XML by the first program and then reload it by the other one. Function EditLogotype edits logotype in that file from which it was loaded by your program (not by TELE). When editing finishes, the function automatically saves changes to that XML file, from which it was loaded last time by the call of LoadLogosAndProfiles function. It is not necessary to reload the file again after that.

Variables

The use of variables dramatically increases TELE2 API capabilities. Analogously to AlphaPro variables, TELE2 ones can be used inside of text lines displayed on TV, and their values are automatically substituted at output. For example, one can make an object of class “static” (see “Object classes” section), and assign the following value to it: “Today is %DATE, Time is: %TIME”. When this object will be shown no screen, current date and time will be automatically displayed. Even more, time will “tick”, i.e. minutes and seconds will automatically change. As it is seen from the sample, variable names must be written in capital letter and the percent sign must go before it. The variables can be created by user – all that should be done for it is to assign a value to the new variable: AssignValue(“MYVAR”, “some text”)

To get a value of a variable one should use the function `GetValue: s=LogoApiObj.GetValue(“MYVAR”)`
In addition to the user-defined variables, there are “system” variables, the values of which are assigned automatically to TELE itself:

```
%TIME - time in format HH:MM:SS
%TIMEHHMM - time in format HH:MM
%DATE - date, for example «Tuesday 3 October 2006»
%SHORTDATE - date in format DD/MM/YY
%THERMO - thermometer indications
```

As it is described in the section “Object classes”, a variable is assigned by default to every upper level object, if it’s initial value is empty. The name of the variable is the same as the name of the object.

General use functions

Sub Login Key as String

This function must be called in the very beginning of a program. Key is the digital part of the HASP key number of TELE-Infochannel software.

Sample: `LogoApiObj.Login(“334”) for key D334`

Functions for work with upper level compositions (logotypes)

Sub AssignValue Var as String, Val as String

Assigns value to a TELE variable. The percent sign is not specified in the name in this case.

Sample: `LogoApiObj.AssignValue(“VAR2”, “text text text”)`

In this sample the value “text text text” is assigned to the variable %VAR2.

The TELE variables may be present in text of objects of **static** class and as a picture file name in **logo** objects. When value of the variable %VAR2 changes, all objects that have ‘%VAR2’ string in their text are updated on screen regarding to the new value of the variable. The other way of changing object values is the function **SendObjDirect**, it assigns the values to the object directly, not by variables.

Function GetValue(Name as String) as String

Returns value of a TELE variable

Sample: `s=LogoApiObj.GetValue(“VAR2”)`

Sub LoadLogosAndProfiles FileName as String

Load logotype (upper level composition) and profile settings from file (XML or earlier TELE 2.0 format)

Sample: `LogoApiObj.LoadLogosAndProfiles(“c:\alphapro\LogosAndProfiles.XML”)`

This sample loads all logotypes and profiles from that file to memory..

Sub EditLogotype Name as String

Starts logotype editor for the logotype Name (upper level composition). When pressing **Save** button in the logotype editor, the current set of compositions and logotypes is saved to the file from which it was loaded (see.

LoadLogosAndProfiles)

Sample: `LogoApiObj.EditLogotype("New Logo1")`

Function ChooseTransition(Transition as string) Name as String

Зарыцкаер Displays *transition* editor. The transitions are used, for example, when a group appers/disappers, and in video-blocks. The settings of transitions are stored in string of this kind:

`"TransitionLen=4 TransitionName=""reveal_right"" Transition=""reveal"" TransitionSoftness=8"`

Transition has 4 parameters: transition length `TransitionLen` (in frames), «softness » of transition

`TransitionSoftness`, transition type `Transition` and transition name `TransitionName`.

The function returns a string having the selected transition.

Sample: `s=LogoApiObj.ChooseTransition("")` ;

Function SendObjDirect Obj as String, Cmd as String

Renews some parameters of the object (this command is used to assign value in real time directly to the logotype/profile object when it is already on screen. Do not use it on 'text' or 'logo' parameters if you use `AssignValue` function with it – it will clear the default variable attached to the object and `AssignValue` will no longer change the object on screen in real time, until you attach the variable to the object again by `SendObjDirect`)

Sample: `LogoApiObj.SendObjDirect("crawl1","addtext=""AAAAAAAAAAAAAAAAAAAA""")`

This sample adds text to the end of a crawl line **crawl1** .

Sample: `LogoApiObj.SendObjDirect("static1","text=""NEWTEXT""")`

Assigning text value to the object of class **static**

Sample: `LogoApiObj.SendObjDirect("logo1","logo=""c:\alphapro\face.logo""")`

Change of picture, displayed in object **logo1**

Sample: `LogoApiObj.SendObjDirect("group","text=""off""")`

Assign value 'off' to a group causes it to disappear from TV screen together with all objects included in it.

Sub SendObjects

Prepares all logotypes and profiles for playback, initializes them. This command must be called in advance, before the playback start, and also after any change made by `EditLogotype`

Sub SendPlayWithLogo(back as String, logo as String, dur as Integer, profile as String)

Starts playback of a video-file of picture back (full path to it is specified, for example `c:\avi\test.avi`, of «clear», if there is no picture), with logotype 'logo' and profile 'profile' having duration dur (in frames). Value (-1) for duration means the endless playback. Value «clear» for filename means that logotypes should overlay on running-through video signal.

Sample: `LogoApiObj.SendPlayWithLogo("clear","New Logo1",10000,"DCS Bars")`

Sample: `LogoApiObj.SendPlayWithLogo("c:\avi\test.avi","New Logo2",1000,"")`

Functions for accessing Alpha Pro

Sub AProAssignValue Name as String, Value as String

Assigns value to AlphaPro variable

The action of the function is equivalent to the `AssignValue` of Alpha Pro API (see. Description of Alpha Pro Extended version).

Sample: `LogoApiObj.AProAssignValue.Login("%VAR1","text text text")`

function AProGetValue(Name as string) as String

Returns value of the AlphaPro variable

The action of the function is equivalent to the `GetValue` of Alpha Pro API (see. Description of Alpha Pro Extended version).

Sample: `s= LogoApiObj.AProGetValue("%VAR1")`

Sub AProExecCommand CmdLine as String

Runs an AlphaPro command

The action of the function is equivalent to ExecCommand of Alpha Pro API (see. Description of Alpha Pro Extended version).

Sub AProGetStatus as String

Returns the status of AlphaPro

The action of the function is equivalent to GetStatus of Alpha Pro API (see. Description of Alpha Pro Extended version).

Functions for accessing text files (for cases when other file functions are not available – as, for example, in DHTML)

Function FileClose FileName as String

Closes file, that was open for reading or writing

Function FileGetLine(FileName as String) as String

Reads a line from file FileName. If it is the first access to the file, it will be automatically opened.

Sub FileWriteLine FileName as String, line as String

Writes the line to file FileName. If it is the first access to the file, it will be automatically created and opened.

Function GetFilePath as String

Opens dialog for selecting file name

Functions for accessing TELE Scheduler schedule

Function AddSchedItem AfterTime as Integer, type as Integer, Name as String

Adds new element to the current schedule:

AfterTime – start time (in frames)

Type – type (0 - clip, 1 - block, 2 - anchor, 3 - empty (name is the duration in this case)

Name - name

Sub DelSchedItem N as Integer

Deletes line number N from the current schedule.

Sub EditBlock Name as String

Starts block editor for block Name

Sub EditClip Name as String

Runs clip editor for clip Name

Sub FindSchedItem time as Integer, Name as String

Find line number in the schedule by its time and name.

Sub PlayBlock Name as String

Starts playback of the block Name

Sub PlayClip Name as String

Starts playback of the clip Name

Sub SetItemLogo n as Integer, Logo as String

Sets logotype for the schedule line

Sub SetItemProfile n as Integer, Profile as String

Sets profile for the schedule line

Sub StartPlaybackFromItem SchedItem as Integer

Starts playback of schedule line number SchedItem

Sub StartPlaybackFromTime Time as Integer

Starts playback of schedule from time Time

Sub StartSystemPlay

Starts playback by system time

Sub StopPlayback

Stops playback

Classes of objects, used in the upper level compositions

Objects have different parameters. The set of parameters depends on class of the object.

The parameters are *variable* and *constant*. The values of constant parameters are same in all logotypes, where the object is present, i.e. it is stored in the object itself. It is easier to set the value of the constant parameter right in the Logotype Editor. The value of variable parameters can be different in different logotypes, because it is stored in logotypes, and such parameters can be changed by AssignValue or SendObjDirect.

The difference between AssignValue and SendObjDirect is that AssignValue assigns value to a variable, and SendObjDirect assigns value directly to the object parameter. It works in the following way: if initially parameter text (for text objects) or logo (for pictures) is empty, i.e. in the Logotype Editor there is no default value, a variable is assigned to it. The name of the variable is the same as the name of the object.

For example, for object mystatic it would be %MYSTATIC. I.e. if we didn't specify a value for 'text' parameter of the mystatic object, the variable %MYSTATIC will be assigned to it automatically. If we assign some value to the %MYSTATIC variable before displaying the object, the object will be displayed with this value. If, after that, that value of the %MYSTATIC be changed (for example, by call of AssignValue function), the object on screen will automatically change (in this sample the static text will change on screen).

But, if the initial value of the parameter was not empty, you could change it only by SendObjDirect call (or you can assign a variable to the object by SendObjDirect and then use AssignValue to change this variable). Analogously, if we have already changed the value of parameter by SendObjDirect, even if the initial value in the Logotype Editor was empty, AssignValue will not work for this object until you assign the variable to it by SendObjDirect.

So, one can put the above information to this table:

AssignValue	SendObjDirect
If used only to set parameters text and logo (for text object – text, for pictures – logo), for group– “on” or “off”.	Can be used for any object parameters
Is convenient for setting initial value, that will be shown when the object is just displayed, but can be used also when the object is already on screen, if the initial value given in the Logotype Editor is empty and SendObjDirect was not used for this parameter of the object	Used only if the object is already on screen
Sample: LogoApiObj.AssignValue ("STATIC1","text text text")	Sample:LogoApiObj.SendObjDirect("static1","text=" "text text text"")

common parameters for all objects:

constant:

x,y,w,h - coordinates
wait (0/1)

variable:

style by default ("helios") - text style
text

object classes:

clock - analogous clock

constant parameters:

clock=path to the folder with clock hands

crawl - crawl line (with pictures)

variable parameters: speed , text
possible sequences in text - \picture \pause \speed
samples:

```
LogoApiObj.SendObjDirect("mycrawl","text=""это текст \picture (C:\avi\pic.tga) pause (100) тоже текст \speed (10)ускоренный текст""")
```

To add text to the end of line:

```
LogoApiObj.SendObjDirect("crawl1","addtext=""AAAAAAAAAAAAAAAAAAAA""")
```

To make this add when the line is already run:

Before every command **SendObjDirect** with addtext parameter

(LogoApiObj.SendObjDirect("crawl1","addtext=""AAAAAAAAAAAAAAAAAAAA""")) one should assign value 1 to the state variable:

variable parameters:
 vspaceing (space between lines),
 Transition - type of transition at appearance/disappearance (reveal_right, reveal_left, crossfade, zoom, fade, wipe)
 TransitionLen - duration of the transition
 TransitionName - name of the transition (path to the picture for wipe)
 TransitionSoftness - «softness» of the transition
 Style (text style),
 text
 Sample:
LogoApiObj.SendObjDirect("mystatic","text=""this is a text"")

Allowable sequences in text - none (only style names)

window - video output in a window

constant parameters:
 scale=[0/1] (scale video or cut it),
 force444=[0/1] - in this mode video file is displayed in window on the background of another video (picture in a picture)
 variable parameters: logo - background picture, resolution is only 720x576 (720x480 for NTSC mode)

group - group(group of objects that appear and disappear together)
 Besides the logotype functions, it makes the objects included in it appear and disappear simultaneously using an effect (transition), set for the group

constant parameters: none
 variable parameters:
 Transition - type of transition at appearance/disappearance (reveal_right, reveal_left, crossfade, zoom, fade, wipe)
 TransitionLen - duration of the transition
 TransitionName - name of the transition (path to the picture for wipe)
 TransitionSoftness - «softness» of the transition
 noanim [0/1] - not take tga files as part of a sequency, even if its name ends with digit
 noloop[0/1] - play animation once and stop on the last frame
 logo="file name" - picture
 text="on"/"off" - show or hide the group
 Sample:
LogoApiObj.SendObjDirect("group1","text=""on"")
 Displays the group together with all objects included in it

LogoApiObj.SendObjDirect("group1","text=""off"")
 Hides the group from screen

Samples

Sample commands for playing video with a logotype:

```
LogoApiObj=CreateObject("TELE.API")
LogoApiObj.Login("334")
LogoApiObj.LoadLogosAndProfiles("c:\alphapro\LogosAndProfiles.XML")
LogoApiObj.SendObjects
LogoApiObj.SendPlayWithLogo("c:\avi\bol.avi","New Logo1",10000,"DSC Bars")
```

Sample commands for displaying video with overlaid logotype with the following change of the static object text:

```
Dim LogoApiObj
Private Sub Form_Load()

LogoApiObj=CreateObject("TELE.API")
LogoApiObj.Login("334")
LogoApiObj.LoadLogosAndProfiles("c:\alphapro\LogosAndProfiles.XML")
LogoApiObj.SendObjects
LogoApiObj.SendPlayWithLogo("clear","New Logo1",-1,"")
End Sub

Private Sub Command1_Click()
LogoApiObj.SendObjDirect("static1","text="""+ Text1.Text)
End Sub
```