eltexalatau

Комплексные решения для построения сетей

Set-Top Box NV-100, NV-102

Спецификация JavaScript API, версия 1.1 (04.2013)

Цифровая телевизионная IP-приставка

Версия документа	Дата выпуска	Содержание изменений
Версия 1.1	04.2013	Добавлены разделы: - 1.2 Управление браузером; - Приложение А Коды кнопок пульта дистанционного управления (ПДУ) в Java Script API; - Приложение В Примеры использования API Изменения в разделах: - 1.6 События; - 2.1 Объект StbDisplay; - 2.2 Объект StbMedia
Версия 1.0	07.2011	Первая публикация
Версия JavaScript API: v1.2		

Целевая аудитория

Спецификация JavaScript API предназначена для разработчиков middleware IPTV приставок. Квалификация специалистов предполагает знание базовых web-технологий: html, javascript, css и практические навыки работы с командной строкой linux.

Условные обозначения

Обозначение	Описание
Полужирный шрифт	Полужирным шрифтом выделены примечания и предупреждения, название глав, заголовков, заголовков таблиц.
Курсив Calibri	Курсивом Calibri указывается информация, требующая особого внимания.
<>	Клавиши клавиатуры

Примечания и предупреждения

Примечания содержат важную информацию, советы или рекомендации по использованию и настройке устройства.

Предупреждения информируют пользователя о ситуациях, которые могут нанести вред устройству или человеку, привести к некорректной работе устройства или потере данных.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

1.1 Аннотация

В данном руководстве приводится подробное описание JavaScript API для приставок производства Eltex.

JavaScript API представляет собой набор объектов JavaScript, которые доступны в специальном браузере (Qt 4.7.0/Webkit). JavaScript API базируется на C API для работы с плеером и параметрами экрана.

1.2 Управление браузером

Существует два варианта использования браузера.

- Первый вариант предполагает, что на приставке отсутствуют другие приложения, есть только браузер. Это означает, что возможно запускать только одно javascript приложение.
- Второй вариант позволяет добавить приложение на javascript, которое выполняется в браузере, в главное меню приставки. Тогда количество javascript приложений неограниченно.

Браузер можно использовать как платформу для любого количества html/javascript приложений. Для этого нужно в */sdk/qt/STBGUI_PLUGIN/* создать директорию с уникальным именем (например, openmw101) и положить в нее бинарный файл браузера *libstbbrowser.so* и файлы настройки по умолчанию: *PluginManifest.xml*, иконки для главного меню. После добавления *libstbbrowser.so* и файла *PluginManifest.xml* приставка распознает новое приложение и отобразит его в главном меню. URL, на который заходит браузер, задаётся один раз в файле манифеста и из JavaScript API не может быть изменён.

Пример файла PluginManifest.xml:

```
  <!DOCTYPE eltex_nv101_plugin_configuration>
<plugin_config>
    <image type="big_icon" src="big_icon.png"/>
    <image type="small_icon" src="small_icon.png"/>
    <text type="brief_descr">   Имя, которое будет отображаться в главном меню</text>
    <text type="long_descr">    Описание приложения </text>
    <version type="plugin_version">version</version>
    <url type="homepage" src="http://demopage.local"/>
    <size width="1280" height="720" type="resolution"/>
</plugin_config>
```

Если браузер является единственным приложением на приставке, то стартовый URL задаётся в файле /home/user/index.htm

```
  <head>
  </head>
  <body>
    <script type="text/javascript">
      window.location = "http://www.google.com/"
    </script>
  </body>
```

Для удобства разработки рекомендуется подключить USB-клавиатуру. Чтобы ввести новый URL, нужно нажать на клавиатуре <Ctrl+L>. Чтобы обновить страницу нужно нажать <Ctrl+R> или *Reload* из меню правой кнопки мыши.

В браузере предоставляются следующие средства отладки:

- WebInspector – встроенный в Webkit отладчик, который позволяет интерактивно искать ошибки в javascript-коде. WebInspector можно вызвать по правой кнопке мыши.
- StbDisplay.Log("line") – печатает отладочную строку в /tmp/jsbrowser.log.
- StbDisplay.Debug("line") – печатает отладочную строку в COM-порт.

1.3 Управление плеером

Для начала воспроизведения контента нужно создать экземпляр плеера, вызвав *id = StbMedia.CreatePlayer()*. Далее этот *id* можно передавать в разные функции *start/stop/pause*, *speed*, *volume*, *viewport* и т.д. Удалить экземпляр плеера можно вызовом *StbMedia.ClosePlayer(id).* Узнать актуальную информацию о воспроизводимом файле можно только после события *ELT_PLAYER_METADATA_UPDATED*. Тип воспроизводимого файла распознается автоматически. Если вызвать *StbMedia.PlayFile(id, path, pos, duration, fps, subt)* с заданной частотой кадров (далее FPS), то будет получено меньше метаданных о файле, так как детальный анализ файла будет пропущен. Когда плеер начинает воспроизведение, порт HDMI переконфигурируется под частоту кадров видеоконтента. Это выглядит как отключение экрана в течение пары секунд. Чтобы получать актуальную информацию о воспроизведении, нужно обрабатывать javascript-события, инициируемые JavaScript API.

1.4 Управление экраном

Слой, в котором отображается интерфейс пользователя, имеет следующие настройки: альфа канал, прозрачный цвет. С помощью этих настроек можно управлять прозрачностью пользовательского интерфейса, вырезать области, закрашенные прозрачным цветом, а также полностью скрывать слой с графическим интерфейсом.

1.5 Управление системными настройками

Буквы русского, английского алфавита и цифры можно вводить, используя пульт приставки. Для переключения раскладки нужно вызвать соответствующую функцию API.

1.6 События

Работа с API ориентирована на события. Большинство вызовов ставят команду в очередь и возвращают управление. О завершении работы этих команд можно узнать, добавив обработчик соответствующего события (Таблица 1).

- ELT_PLAYER_STATUS_CHANGED – состояние плеера изменилось;
- ELT_PLAYER_METADATA_UPDATED – новые метаданные о видео получены;
- ELT_PLAYER_POSITION_CHANGED – перемещение на новую позицию завершено;
- ELT_PLAYER_AUDIO_TRACK_SWITCHED – переключение аудио дорожки завершено;
- ELT_PLAYER_VIDEO_TRACK_SWITCHED – переключение видео дорожки завершено;
- ELT_PLAYER_SUBTL_TRACK_SWITCHED – переключение дорожки субтитров завершено;
- ELT_PLAYER_VOLUME_CHANGED – громкость изменилась;

- ELT_PLAYER_ZOOM_CHANGED – зум изменился;
- ELT_PLAYER_MUTE_CHANGED – звук отключился или включился;
- ELT_PLAYER_MULTICAST_GROUP_SWITCHED – произошло изменение состояния в протоколе IGMP.

Таблица 1 – Номера событий

Событие	**Номер события**
var ELT_PLAYER_STATUS_CHANGED	1
var ELT_PLAYER_METADATA_UPDATED	2
var ELT_PLAYER_POSITION_CHANGED	3
var ELT_PLAYER_AUDIO_TRACK_SWITCHED	4
var ELT_PLAYER_VIDEO_TRACK_SWITCHED	5
var ELT_PLAYER_SUBTL_TRACK_SWITCHED	6
var ELT_PLAYER_VOLUME_CHANGED	7
var ELT_PLAYER_ZOOM_CHANGED	8
var ELT_PLAYER_MUTE_CHANGED	9
var ELT_PLAYER_MULTICAST_GROUP_SWITCHED	11
var ELT_OUTPORTS_STATE	129

Когда плеер начинает воспроизведение нового контента, начинают срабатывать описанные события. Это означает, что плеер разобрал поток и прочитал из него данную информацию.

Плеер может находиться в следующих состояниях:

- ERROR — плеер не может воспроизвести запрашиваемый контент;
- LOADING — плеер начинает воспроизведение (например, наполняет кэш);
- PAUSED — плеер приостановлен;
- STOPPED — плеер остановлен;
- MOVING — плеер в процессе перехода на новую позицию;
- END_OF_STREAM — плеер дошёл до конца файла или конца потока;
- и другие события, описывающие скорость воспроизведения.

2 ОПИСАНИЕ ОБЪЕКТОВ

2.1 Объект StbDisplay

Содержит функции, не относящиеся к воспроизведению медиаконтента. Список атрибутов и методов объекта StbDisplay приведен в таблице 2.

Таблица 2 – Список атрибутов и методов объекта StbDisplay

Атрибуты	
<static>	StbDisplay.APIVersion Версия JavaScript API.
<static>	StbDisplay.DeviceModel Модель устройства.
<static>	StbDisplay.EthernetHardwareAddress MAC-адрес сетевого интерфейса.
<static>	StbDisplay.FirmwareVersion Версия программного обеспечения.
<static>	StbDisplay.IpAddress IP-адрес сетевого интерфейса.
<static>	StbDisplay.SerialNumber Серийный номер устройства.
Методы	
<static>	StbDisplay.Debug(debug) Выводит строку в последовательный порт.
<static>	StbDisplay.ExecBlocking(scriptname, args) Функция исполняет скрипт из директории /sdk/custom_scripts/.
<static>	StbDisplay.ExitToMenu() Полностью закрывает браузер middleware и выходит в главное меню приставки.
<static>	StbDisplay.GetAlphaLevel() Возвращает уровень прозрачности пользовательского интерфейса.
<static>	StbDisplay.GetBootParamValue(key) Возвращает загрузочный параметр по ключу.
<static>	StbDisplay.GetChromaKey() Возвращает прозрачный цвет.
<static>	StbDisplay.GetCurrentRMLayout() Возвращает раскладку пульта.
<static>	StbDisplay.GetGraphicsViewport() Возвращает размеры области вывода пользовательского интерфейса.

<static>	StbDisplay.IsChromaKeyEnable() Возвращает состояние режима прозрачного цвета.
<static>	StbDisplay.IsGraphicsVisible() Позволяет узнать, скрыт пользовательский интерфейс или отображается на экране.
<static>	StbDisplay.IsOutportsEnabled() Проверяет состояние видео выхода (HDMI).
<static>	StbDisplay.ListDirs(directory) Возвращает список директорий с внешнего носителя или сетевого ресурса.
<static>	StbDisplay.ListFiles(directory) Возвращает список файлов с внешнего носителя или сетевого ресурса.
<static>	StbDisplay.Log(log) Выводит строку в файл /tmp/jsbrowser.log.
<static>	StbDisplay.Reboot() Перезагружает приставку.
<static>	StbDisplay.SetAlphaLevel(alpha) Устанавливает уровень прозрачности пользовательского интерфейса.
<static>	StbDisplay.SetBootParamValue(key, value) Управляет загрузочными параметрами приставки в формате ключ-значение.
<static>	StbDisplay.SetChromaKey(color) Устанавливает прозрачный цвет.
<static>	StbDisplay.SetChromaKeyEnable(enable) Включает или выключает режим прозрачного цвета.
<static>	StbDisplay.SetCurrentRMLayout(layout) Позволяет выбрать раскладку пульта.
<static>	StbDisplay.SetGraphicsViewport(x, y, w, h) Устанавливает область вывода пользовательского интерфейса.
<static>	StbDisplay.SetGraphicsVisible(visibility) Позволяет скрыть или отобразить пользовательский интерфейс.
<static>	StbDisplay.SetKeyEventPeriod(period) Устанавливает период времени, в течение которого будут фильтроваться одинаковые клавиши пульта.
<static>	StbDisplay.SwitchToNextRMLayout() Переключает раскладку пульта на следующую: DIG-ENG-RUS.
<static>	StbDisplay.UpgradeFirmware(firmware) Ставит в очередь прошивку по HTTP-ссылке.

2.1.1 Атрибуты объекта StbDisplay

StbDisplay.APIVersion

<static> *StbDisplay*.APIVersion

Версия JavaScript API.

StbDisplay.DeviceModel

<static> *StbDisplay*.DeviceModel

Модель устройства.

StbDisplay.EthernetHardwareAddress

<static> *StbDisplay*.EthernetHardwareAddress

MAC-адрес сетевого интерфейса.

StbDisplay.FirmwareVersion

<static> *StbDisplay*.FirmwareVersion

Версия программного обеспечения.

StbDisplay.IpAddress

<static> *StbDisplay*.IpAddress

IP-адрес сетевого интерфейса.

StbDisplay.SerialNumber

<static> *StbDisplay.SerialNumber*

Серийный номер устройства.

2.1.2 Методы объекта StbDisplay

StbDisplay.Debug

```
<static> StbDisplay.Debug(debug)
```

Выводит строку в последовательный порт.

Параметры:

Параметры	Значения	Описание
debug		строка

Возвращаемое значение:

Статус операции.

Пример:

```
StbDisplay.Debug("debug message");
```

StbDisplay.ExecBlocking

```
<static> StbDisplay.ExecBlocking(scriptname, args)
```

Функция исполняет скрипт из директории /sdk/custom_scripts/. Интерпретатор javascript будет приостановлен на время выполнения скрипта.

Параметры:

Параметры	Значения	Описание
scriptname		имя скрипта
args		аргументы

Возвращаемое значение:

Объект:

```
{
   "status" : "OK",
   "exit"   : 0,
   "stdout" : ["line","line",...],
   "stderr" : ["line","line",...]
}
```

Или объект с ошибкой:

```
{
   "status"  : "FAIL",
   "code"    : 40,
   "message" : "error description"

}
```

Пример:

```
var result = StbDisplay.ExecBlocking("example.sh", "-la");

 if(result.status == "OK"){
        alert(result.exit);
        alert(result.stdout);
        alert(result.stderr);
 } else if (result.status == "FAIL"){
        alert(result.status);
        alert(result.message);
        alert(result.code);
 }
```

StbDisplay.ExitToMenu

```
<static> StbDisplay.ExitToMenu()
```

Полностью закрывает браузер middleware и выходит в главное меню приставки. Перед вызовом функции нужно освободить все захваченные ресурсы, вернуть начальные значения параметров прозрачного цвета и альфа-канала.

Возвращаемое значение:

Статус операции.

Пример:

```
var RC_RETURN_KEY = 27;
...
switch(code) {
 ...
 case RC_RETURN_KEY:
        alert("return");
         StbDisplay.ExitToMenu();
         break
...
}
```

StbDisplay.GetAlphaLevel

```
<static> StbDisplay.GetAlphaLevel()
```

Возвращает уровень прозрачности пользовательского интерфейса.

Возвращаемое значение:

Уровень прозрачности (0-255).

Пример:

```
var alpha = StbDisplay.GetAlphaLevel();
StbDisplay.SetAlphaLevel(alpha/2);
```

StbDisplay.GetBootParamValue

```
<static> StbDisplay.GetBootParamValue(key)
```

Возвращает загрузочный параметр по ключу. Приставка применяет эти параметры при загрузке. Поддерживается следующий ключ : "dfbres" - разрешение пользовательского интерфейса.

Параметры:

Параметры	Значения	Описание
key	dfbres	ключ

Возвращаемое значение:

Значение загрузочного параметра.

Пример:

```
var value = StbDisplay.GetBootParamValue("dfbres");
```

StbDisplay.GetChromaKey

```
<static> StbDisplay.GetChromaKey()
```

Возвращает прозрачный цвет.

Возвращаемое значение:

Прозрачный цвет в формате RGB.

Пример:

```
var color = StbDisplay.GetChromaKey();
var string = color.toString(16); //"FF00EE"
```

StbDisplay.GetCurrentRMLayout

```
<static> StbDisplay.GetCurrentRMLayout()
```

Возвращает раскладку пульта.

Возвращаемое значение:

DIG = 0 - цифры
ENG = 1 - английские буквы
RU = 2 - русские буквы

StbDisplay.GetGraphicsViewport

```
<static> StbDisplay.GetGraphicsViewport()
```

Возвращает размеры области вывода пользовательского интерфейса.

Возвращаемое значение:

Объект:

```
{
   "x":"x",
   "y":"y",
   "width":"ширина",
   "height":"высота"
}
```

Пример:

```
 var viewport = StbDisplay.GetGraphicsViewport();

 alert(viewport.x);
 alert(viewport.y);
 alert(viewport.width);
 alert(viewport.height);
```

StbDisplay.IsChromaKeyEnable

```
<static> StbDisplay.IsChromaKeyEnable()
```

Возвращает состояние режима прозрачного цвета.

Возвращаемое значение:

Состояние прозрачного цвета:

- 0 - выключен;
- 1 - включен.

Пример:

```
if (StbDisplay.IsChromaKeyEnable() == 0) {
        StbDisplay.SetChromaKeyEnable(1);
 }
```

StbDisplay.IsGraphicsVisible

```
<static> StbDisplay.IsGraphicsVisible()
```

Позволяет узнать, скрыт пользовательский интерфейс или отображается на экране.

Возвращаемое значение:

Состояние:

- 0 - пользовательский интерфейс скрыт;
- 1 - пользовательский интерфейс отображается.

StbDisplay.IsOutportsEnabled

```
<static> StbDisplay.IsOutportsEnabled()
```

Проверяет состояние видео выхода (HDMI).

Возвращаемое значение:

Состояние:

- 0 — выключен;
- 1 — включен.

Пример:

```
StbMedia.onEvent = "onMediaEvent();";
function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);
    switch (event_type) {
    case ELT_OUTPORTS_STATE:
              var outports = StbDisplay.IsOutportsEnabled();
        alert(outports);
              break;
      }
}
```

StbDisplay.ListDirs

```
<static> StbDisplay.ListDirs(directory)
```

Возвращает список директорий с внешнего носителя или сетевого ресурса.

Параметры:

Параметры	Значения	Описание
directory		имя директории, "/" - корневая директория

Возвращаемое значение:

Массив-список директорий.

```
[
  "dir1",
  "dir2"
]
```

Пример:

```
function listDirs()
{
 var dir = StbDisplay.ListDirs("/");
 alert(dir);
 if(dir.status == "FAIL") {
        alert(fail);
        return;
 }
 alert(dir.length);
 for(var i = 0; i<dir.length; i++){
        alert(dir[i]);
 }
}
```

StbDisplay.ListFiles

```
<static> StbDisplay.ListFiles(directory)
```

Возвращает список файлов с внешнего носителя или сетевого ресурса.

Параметры:

Параметры	Значения	Описание
directory		имя директории, "/" - корневая директория

Возвращаемое значение:

Массив-список файлов.

```
[
  "file1",
  "file2"
]
```

Пример:

```
function listFiles()
{
var files = StbDisplay.ListFiles("/local/usbDisk0");
 alert(files);
 if(files.status == "FAIL") {
        alert(fail);
        return;
 }
 alert(files.length);
```

```
  for(var i = 0; i<files.length; i++){
        var esc = escape(files[i]);
        var dec = decodeURIComponent(esc);
        alert(dec);
  }
}
```

StbDisplay.Log

```
<static> StbDisplay.Log(log)
```

Выводит строку в файл /tmp/jsbrowser.log.
Размер файла 1Мб.

Параметры:

Параметры	Значения	Описание
log		строка

Возвращаемое значение:

Статус операции.

Пример:

```
StbDisplay.Log("log message");
```

StbDisplay.Reboot

```
<static> StbDisplay.Reboot()
```

Функция перезагружает приставку.

Возвращаемое значение:

Статус операции.

StbDisplay.SetAlphaLevel

```
<static> StbDisplay.SetAlphaLevel(alpha)
```

Устанавливает уровень прозрачности пользовательского интерфейса.

Параметры:

Параметры	Значения	Описание
alpha	0 - 255 0 - полностью прозрачный	уровень прозрачности

Возвращаемое значение:

Статус операции.

Пример:

```
var alpha = 100;
while(alpha < 200){
 sleep(1);
 StbDisplay.SetAlphaLevel(alpha);
 alpha++;}
```

StbDisplay.SetBootParamValue

```
<static> StbDisplay.SetBootParamValue(key, value)
```

Управляет загрузочными параметрами приставки в формате ключ-значение. Приставка применяет эти параметры при загрузке. Настройки сохраняются на внутреннюю флэш-память.

Поддерживается следующий ключ:

"dfbres" - разрешение пользовательского интерфейса. Не используйте нестандартные разрешение "dfbres" (например 836x351) - это может вызвать нестабильную работу приставки.

Примечание:

Как описано в пункте 1.2, каждый плагин складывается в индивидуальную папку, имя этой папки является идентификатором плагина. То есть настройки сохраняются для каждого плагина отдельно в директорию /home/user/config/plugins/pluginname/PluginManifest.xml. Если браузер является единственным приложением на приставке, то изменяется файл /etc/settings.xml.

Параметры:

Параметры	Значения	Описание
key	dfbres	ключ
value		значение

Возвращаемое значение:

Статус операции.

Пример:

```
StbDisplay.SetBootParamValue("dfbres", "1920x1080");
```

StbDisplay.SetChromaKey

```
<static> StbDisplay.SetChromaKey(color)
```

Устанавливает прозрачный цвет.

Параметры:

Параметры	Значения	Описание
color		прозрачный цвет в целочисленном виде

Возвращаемое значение:

Статус операции.

Пример:

```
var color = "FF00EE";
StbDisplay.SetChromaKey(parseInt(color,16));
```

StbDisplay.SetChromaKeyEnable

```
<static> StbDisplay.SetChromaKeyEnable(enable)
```

Включает или выключает режим прозрачного цвета

Параметры:

Параметры	Значения	Описание
enable	0 – выключить 1 - включить	режим прозрачного цвета

Возвращаемое значение:

Статус операции.

Пример:

```
if (StbDisplay.IsChromaKeyEnable() == 0) {
        StbDisplay.SetChromaKeyEnable(1);
 }
```

StbDisplay.SetCurrentRMLayout

```
<static> StbDisplay.SetCurrentRMLayout(layout)
```

Позволяет выбрать раскладку пульта.

Параметры:

Параметры	Значения	Описание
layout	DIG = 0 - цифры ENG = 1 - английские буквы RU = 2 - русские буквы	раскладка пульта

Возвращаемое значение:

Установленная раскладка.

StbDisplay.SetGraphicsViewport

```
<static> StbDisplay.SetGraphicsViewport(x, y, w, h)
```

Устанавливает область вывода пользовательского интерфейса.

Параметры:

Параметры	Значения	Описание
x		координата x
y		координата y
w		ширина
h		высота

Возвращаемое значение:

Статус операции.

StbDisplay.SetGraphicsVisible

```
<static> StbDisplay.SetGraphicsVisible(visibility)
```

Позволяет скрыть или отобразить пользовательский интерфейс.

Параметры:

Параметры	Значения	Описание
visibility	0 - скрыть пользовательский интерфейс 1 - отобразить пользовательский интерфейс	режим отображения

Возвращаемое значение:

Статус операции.

Пример:

```
if (StbDisplay.IsGraphicsVisible() == 0) {
        StbDisplay.SetGraphicsVisible(1);
 }
```

StbDisplay.SetKeyEventPeriod

```
<static> StbDisplay.SetKeyEventPeriod(period)
```

Устанавливает период времени, в течение которого будут фильтроваться одинаковые клавиши пульта.

Параметры:

Параметры	Значения	Описание
period		период в миллисекундах

Возвращаемое значение:

Статус операции.

StbDisplay.SwitchToNextRMLayout

```
<static> StbDisplay.SwitchToNextRMLayout()
```

Переключает раскладку пульта на следующую: DIG-ENG-RUS.

Возвращаемое значение:

Следующая раскладка.

StbDisplay.UpgradeFirmware

```
<static> StbDisplay.UpgradeFirmware(firmware)
```

Ставит в очередь прошивку по HTTP-ссылке. При следующей загрузке приставки прошивка будет обновлена. Это обновление принудительное, можно обновиться на устаревшую версию или версию с невалидной цифровой подписью.

Параметры:

Параметры	Значения	Описание
firmware		адрес прошивки (http://localhost/nv101img_120327_0.415.22)

Возвращаемое значение:

Статус операции.

Пример:

```
StbDisplay.UpgradeFirmware("http://localhost/nv101img_120327_0.415.22");
StbDisplay.Reboot();
```

2.2 Объект StbMedia

Содержит функции, которые относятся к воспроизведению медиаконтента. Список атрибутов и методов объекта StbMedia приведен в таблице 3.

Таблица 3 – Список атрибутов и методов объекта StbMedia

Атрибуты	
<static>	StbMedia.onEvent Устанавливает функцию обратного вызова в текстовом формате.
Методы	
<static>	StbMedia.ClosePlayer(id) Удаляет экземпляр плеера.
<static>	StbMedia.Continue(id) Возобновляет воспроизведение после паузы.
<static>	StbMedia.CreatePlayer() Создаёт новый экземпляр плеера.
<static>	StbMedia.GetAudioPid(id) Возвращает номер (PID) аудиодорожки.
<static>	StbMedia.GetAudioTrackList(id) Возвращает список доступных аудиодорожек.
<static>	StbMedia.GetDuration(id) Возвращает длительность файла.
<static>	StbMedia.GetEventDescription(id) Возвращает описание последнего возникшего события.
<static>	StbMedia.GetFps(id) Возвращает частоту кадров (FPS) для контента.
<static>	StbMedia.GetLastMulticastEvent(id) Возвращает последнее IGMP-событие для плеера.
<static>	StbMedia.GetPlayerAlphaLevel(id) Возвращает уровень прозрачности плеера.
<static>	StbMedia.GetPlayerDisplayMode(id) Возвращает режим отображения видео.
<static>	StbMedia.GetPlayersCount() Возвращает количество созданных экземпляров плеера.
<static>	StbMedia.GetPlayerState(id) Возвращает состояние плеера.
<static>	StbMedia.GetPlayerViewPort(id) Возвращает параметры прямоугольной области, в которую вписан плеер.

<static>	StbMedia.GetPosition(id) Возвращает текущую позицию.
<static>	StbMedia.GetSpeed(id) Возвращает скорость воспроизведения.
<static>	StbMedia.GetSubtitlePid(id) Возвращает PID дорожки субтитров.
<static>	StbMedia.GetSubtitleTrackList(id) Возвращает список доступных дорожек субтитров.
<static>	StbMedia.GetVideoPid(id) Возвращает номер (PID) видеодорожки.
<static>	StbMedia.GetVideoTrackList(id) Возвращает список доступных видеодорожек.
<static>	StbMedia.GetVolume(id) Возвращает уровень громкости.
<static>	StbMedia.GetZoom(id) Возвращает коэффициент зуммирования.
<static>	StbMedia.IsMute(id) Возвращает состояние режима "без звука".
<static>	StbMedia.IsPlayerVisible(id) Позволяет узнать отображается плеер или нет.
<static>	StbMedia.IsSubtitlesVisible(id) Позволяет узнать отображаются субтитры или нет.
<static>	StbMedia.NextFrame(id) Переводит плеер на следующий кадр.
<static>	StbMedia.Pause(id) Ставит плеер на паузу.
<static>	StbMedia.PlayFile(id, path, pos, duration, fps, subt) Запускает файл на воспроизведение.
<static>	StbMedia.PlayHttp(id, path, opt) Запускает на воспроизведение контент по HTTP-ссылке.
<static>	StbMedia.PlayMagnet(id, magnet, name, size, pos, duration, fps) Запускает на проигрывание файл по magnet-ссылке.
<static>	StbMedia.PlayRtsp(id, path, mode) Запускает воспроизведение по RTSP-ссылке.
<static>	StbMedia.PlayStream(id, path) Воспроизвести мультикастовый поток.
<static>	StbMedia.SetAudioPid(id, pid)

	Выбирает аудиодорожку.
<static>	StbMedia.SetMute(id, mute) Устанавливает режим "без звука".
<static>	StbMedia.SetPlayerAlphaLevel(id, alpha) Устанавливает уровень прозрачности плеера.
<static>	StbMedia.SetPlayerDisplayMode(id, mode) Устанавливает режим отображения видео.
<static>	StbMedia.SetPlayerViewPort(id, x, y, w, h) Вписывает плеер в прямоугольную область.
<static>	StbMedia.SetPlayerVisible(id, visibility) Может скрыть плеер с экрана, не останавливая воспроизведение, или вернуть плеер обратно на экран.
<static>	StbMedia.SetPosition(id, pos) Установить позицию в миллисекундах.
<static>	StbMedia.SetSpeed(id, speed) Устанавливает скорость воспроизведения.
<static>	StbMedia.SetSubtitlePid(id, pid) Выбирает дорожку субтитров.
<static>	StbMedia.SetSubtitlesVisible(id, visibility) Позволяет скрыть субтитры с экрана.
<static>	StbMedia.SetVolume(id, vol) Устанавливает уровень громкости.
<static>	StbMedia.SetZoom(id, zoom) Устанавливает коэффициент зуммирования.
<static>	StbMedia.Stop(id) Останавливает воспроизведение.

2.2.1 Атрибуты объекта StbMedia

StbMedia.onEvent

<static> *StbMedia*.onEvent

Устанавливает функцию обратного вызова в текстовом формате.

```
StbMedia.onEvent = "onMediaEvent();";
function onMediaEvent() {
 var event = StbMedia.GetEventDescription();
 var event_type = parseInt(event.type);

 switch (event_type) {
 case ELT_PLAYER_STATUS_CHANGED:
     break;
 case ELT_PLAYER_METADATA_UPDATED:
     break;
 case ELT_PLAYER_POSITION_CHANGED:
     break;
 case ELT_PLAYER_AUDIO_TRACK_SWITCHED:
     break;
 case ELT_PLAYER_VIDEO_TRACK_SWITCHED:
     break;
 case ELT_PLAYER_SUBTL_TRACK_SWITCHED:
     break;
 case ELT_PLAYER_VOLUME_CHANGED:
     break;
 case ELT_PLAYER_ZOOM_CHANGED:
     break;
 case ELT_PLAYER_MUTE_CHANGED:
     break;
 case ELT_PLAYER_MULTICAST_GROUP_SWITCHED:
     break;
}}
```

2.2.2 Методы объекта StbMedia

StbMedia.ClosePlayer

```
<static> StbMedia.ClosePlayer(id)
```

Удаляет экземпляр плеера.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Статус операции.

Пример:

```
var id = StbMedia.CreatePlayer();
StbMedia.ClosePlayer(id);
```

StbMedia.Continue

```
<static> StbMedia.Continue(id)
```

Возобновляет воспроизведение после паузы.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Статус операции.

Пример:

```
function goContinue(){

 StbMedia.Continue(id)
;
}

StbMedia.onEvent = "onMediaEvent();";
   function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);

    switch (event_type) {
    case ELT_PLAYER_STATUS_CHANGED:
        var state = StbMedia.GetPlayerState(id);
        if(state == PAUSED){
```

```
                setTimeout(goContinue, 3000);
            }
            break;
        ...
          }
}
var id = StbMedia.CreatePlayer();
... //запускаем на воспроизведение, ставим на паузу
```

StbMedia.CreatePlayer

```
<static> StbMedia.CreatePlayer()
```

Создаёт новый экземпляр плеера.

Возвращаемое значение:

Идентификатор созданного плеера или отрицательное число в случае ошибки.

Пример:

```
var id = StbMedia.CreatePlayer();
```

StbMedia.GetAudioPid

```
<static> StbMedia.GetAudioPid(id)
```

Возвращает номер (PID) аудиодорожки.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

PID аудиодорожки.

StbMedia.GetAudioTrackList

```
<static> StbMedia.GetAudioTrackList(id)
```

Возвращает список доступных аудиодорожек. Функцию можно вызывать только по событию ELT_PLAYER_METADATA_UPDATED.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Список аудиодорожек в формате JSON.

```
[
   {id:id, descr:descr, supported:supported},
   {id:id, descr:descr, supported:supported}
]
```

Пример:

```
StbMedia.onEvent = "onMediaEvent();";
function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);
    switch (event_type) {
    case ELT_PLAYER_METADATA_UPDATED:
              var vp = StbMedia.GetAudioTrackList(id);
        alert(vp.length);
        alert(vp[0].id);
        alert(vp[0].descr);
        alert(vp[0].supported);
        break;
    ...
      }
}
```

StbMedia.GetDuration

```
<static> StbMedia.GetDuration(id)
```

Возвращает длительность файла. Эту функцию можно вызывать только после события ELT_PLAYER_METADATA_UPDATED.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Длительность контента в миллисекундах.

Пример:

```
StbMedia.onEvent = "onMediaEvent();";
 function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);
    switch (event_type) {
    case ELT_PLAYER_METADATA_UPDATED:
        StbMedia.SetPosition(id, StbMedia.GetDuration(id)/2);
        break;
    ...
      }
}
```

StbMedia.GetEventDescription

```
<static> StbMedia.GetEventDescription()
```

Возвращает описание последнего возникшего события.

Возвращаемое значение:

Объект {"id":"id","type":"type"}:

- id - идентификатор плеера;
- type - тип события.

Список событий:

```
var ELT_PLAYER_STATUS_CHANGED = 1;
var ELT_PLAYER_METADATA_UPDATED = 2;
var ELT_PLAYER_POSITION_CHANGED = 3;
var ELT_PLAYER_AUDIO_TRACK_SWITCHED = 4;
var ELT_PLAYER_VIDEO_TRACK_SWITCHED = 5;
var ELT_PLAYER_SUBTL_TRACK_SWITCHED = 6;
var ELT_PLAYER_VOLUME_CHANGED = 7;
var ELT_PLAYER_ZOOM_CHANGED = 8;
var ELT_PLAYER_MUTE_CHANGED = 9;
var ELT_PLAYER_MULTICAST_GROUP_SWITCHED = 11;
```

Пример:

```
var event = StbMedia.GetEventDescription();
var event_type = parseInt(event.type);
```

StbMedia.GetFps

```
<static> StbMedia.GetFps(id)
```

Возвращает частоту кадров (FPS) для контента.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

FPS.

StbMedia.GetLastMulticastEvent

```
<static> StbMedia.GetLastMulticastEvent(id)
```

Возвращает последнее IGMP-событие для плеера. Вход и выход из мультикастовой группы.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Объект

```
{
   "ip_port":"226.0.0.2:213",
   "state"  :"join"
}
```

Пример:

```
var mc = StbMedia.GetLastMulticastEvent(event.id);
alert("ip-port " + mc.ip_port + " state " + mc.state + "\n");
```

StbMedia.GetPlayerAlphaLevel

```
<static> StbMedia.GetPlayerAlphaLevel(id)
```

Возвращает уровень прозрачности плеера.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Уровень прозрачности (0-255), где 0 - полностью прозрачный.

StbMedia.GetPlayerDisplayMode

```
<static> StbMedia.GetPlayerDisplayMode(id)
```

Возвращает режим отображения видео.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Режим отображения:
- нормальный (DISPLAY NORMAL=1);
- обрезанный по краям (DISPLAY CROP=2);
- растянутый (DISPLAY STRETCH=3).

Пример:

```
var mode = StbMedia.GetPlayerDisplayMode(id);
switch(mode){
       case DISPLAY_NORMAL:
              StbMedia.SetPlayerDisplayMode(id, DISPLAY_CROP);
              break;
       case DISPLAY_CROP:
              StbMedia.SetPlayerDisplayMode(id, DISPLAY_STRETCH);
              break;
       case DISPLAY_STRETCH:
              StbMedia.SetPlayerDisplayMode(id, DISPLAY_NORMAL);
              break;
       default:
              StbMedia.SetPlayerDisplayMode(id, DISPLAY_NORMAL);
}
```

StbMedia.GetPlayersCount

```
<static> StbMedia.GetPlayersCount()
```

Возвращает количество созданных экземпляров плеера.

Возвращаемое значение:

Количество экземпляров плеера.

StbMedia.GetPlayerState

```
<static> StbMedia.GetPlayerState(id)
```

Возвращает состояние плеера.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Состояние плеера.

Пример:

```
    var ELT_SPEED_FWD_1X = 0x1;
    var ELT_SPEED_FWD_2X = 0x2;
    var ELT_SPEED_FWD_4X = 0x4;
```

```
var ELT_SPEED_FWD_8X = 0x8;
var ELT_SPEED_FWD_16X = 0x10;
var ELT_SPEED_FWD_32X = 0x20;
var ELT_SPEED_BWD_1X = 0x101;
var ELT_SPEED_BWD_2X = 0x102;
var ELT_SPEED_BWD_4X = 0x104;
var ELT_SPEED_BWD_8X = 0x108;
var ELT_SPEED_BWD_16X = 0x110;
var ELT_SPEED_BWD_32X = 0x120;

var ERROR = 0x80000000;
var ERROR_LOW_DR = 0x80000001;
var ERROR_NOT_CN = 0x80000002;
var ERROR_NOT_FN = 0x80000004;
var ERROR_CANT_OPEN = 0x80000008;
var ERROR_UNKNOWN_CONTENT = 0x80000010;
var LOADING = 1;
var PAUSED = 2;
var STOPPED = 3;
var MOVING = 4;
var END_OF_STREAM = 5;
var PLAYING_FWD_1x = 0x10000000 | ELT_SPEED_FWD_1X;
var PLAYING_FWD_2X = 0x10000000 | ELT_SPEED_FWD_2X;
var PLAYING_FWD_4X = 0x10000000 | ELT_SPEED_FWD_4X;
var PLAYING_FWD_8X = 0x10000000 | ELT_SPEED_FWD_8X;
var PLAYING_FWD_16X = 0x10000000 | ELT_SPEED_FWD_16X;
var PLAYING_FWD_32X = 0x10000000 | ELT_SPEED_FWD_32X;
var PLAYING_BWD_1X = 0x10000000 | ELT_SPEED_BWD_1X;
var PLAYING_BWD_2X = 0x10000000 | ELT_SPEED_BWD_2X;
var PLAYING_BWD_4X = 0x10000000 | ELT_SPEED_BWD_4X;
var PLAYING_BWD_8X = 0x10000000 | ELT_SPEED_BWD_8X;
var PLAYING_BWD_16X = 0x10000000 | ELT_SPEED_BWD_16X;
var PLAYING_BWD_32X = 0x10000000 | ELT_SPEED_BWD_32X;

var state = StbMedia.GetPlayerState(id);

switch(state){
case PAUSED:
case MOVING:
...
}
```

StbMedia.GetPlayerViewPort

```
<static> StbMedia.GetPlayerViewPort(id)
```

Возвращает параметры прямоугольной области, в которую вписан плеер.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Объект содержащий координаты.

```
{
   "x":"x",
   "y":"y",
   "width":"ширина",
   "height":"высота"
}
```

Пример:

```
 var vp = StbMedia.GetPlayerViewPort(id);

 alert(vp.x);
 alert(vp.y);
 alert(vp.width);
 alert(vp.height);
```

StbMedia.GetPosition

<static> *StbMedia*.GetPosition(id)

Возвращает текущую позицию.
Функцию можно вызывать только по событию ELT_PLAYER_POSITION_CHANGED.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Текущая позиция в миллисекундах.

Пример:

```
StbMedia.onEvent = "onMediaEvent();";
function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);
    switch (event_type) {
    case ELT_PLAYER_POSITION_CHANGED:
              var pos = StbMedia.GetPosition(id);
        alert(pos);
              break;
    ...
      }
}
```

StbMedia.GetSpeed

```
<static> StbMedia.GetSpeed(id)
```

Возвращает скорость воспроизведения.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Одно из возможных значений скорости.

```
var ELT_SPEED_FWD_1X = 0x1;
var ELT_SPEED_FWD_2X = 0x2;
var ELT_SPEED_FWD_4X = 0x4;
var ELT_SPEED_FWD_8X = 0x8;
var ELT_SPEED_FWD_16X = 0x10;
var ELT_SPEED_FWD_32X = 0x20;
var ELT_SPEED_BWD_1X = 0x101;
var ELT_SPEED_BWD_2X = 0x102;
var ELT_SPEED_BWD_4X = 0x104;
var ELT_SPEED_BWD_8X = 0x108;
var ELT_SPEED_BWD_16X = 0x110;
var ELT_SPEED_BWD_32X = 0x120;
```

Пример:

```
StbMedia.onEvent = "onMediaEvent();";
function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);
    switch (event_type) {
    case ELT_PLAYER_STATUS_CHANGED:
              speed = StbMedia.GetSpeed(event.id);
        alert(speed);
              break;
    ...
      }
}
```

StbMedia.GetSubtitlePid

```
<static> StbMedia.GetSubtitlePid(id)
```

Возвращает PID дорожки субтитров.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

PID дорожки субтитров.

StbMedia.GetSubtitleTrackList

```
<static> StbMedia.GetSubtitleTrackList(id)
```

Возвращает список доступных дорожек субтитров.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Объект-список субтитров.

```
[
  {id:id, descr:descr, supported:supported, encoding:encoding},
  {id:id, descr:descr, supported:supported, encoding:encoding}
]
```

- id – PID дорожки;
- descr – строка описания дорожки;
- supported – int. 1 – поддерживается, 0 – не поддерживается.

Пример:

```
var vp = StbMedia.GetSubtitleTrackList(id);

 alert(vp.length);
 alert(vp[0].id);
 alert(vp[0].descr);
 alert(vp[0].supported);
}
```

StbMedia.GetVideoPid

```
<static> StbMedia.GetVideoPid(id)
```

Возвращает номер (PID) видеодорожки.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

PID видеодорожки.

StbMedia.GetVideoTrackList

```
<static> StbMedia.GetVideoTrackList(id)
```

Возвращает список доступных видеодорожек. Функцию можно вызывать только по событию ELT_PLAYER_METADATA_UPDATED.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Массив объектов видеодорожек.

```
[
 {id:id ,descr:descr,supported:supported},
 {id:id ,descr:descr,supported:supported},
 ...]
```

- id – PID дорожки;
- descr – строка описания дорожки;
- supported – int. 1 – поддерживается, 0 – не поддерживается.

Пример:

```
StbMedia.onEvent = "onMediaEvent();";
function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);
    switch (event_type) {
    case ELT_PLAYER_METADATA_UPDATED:
              var vp = StbMedia.GetVideoTrackList(id);
        alert(vp.length);
        alert(vp[0].id);
        alert(vp[0].descr);
        alert(vp[0].supported);
        break;
   ...
      }}
```

StbMedia.GetVolume

```
<static> StbMedia.GetVolume(id)
```

Возвращает уровень громкости.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Уровень громкости.

Пример:

```
StbMedia.SetVolume(StbMedia.GetVolume(id)+1);
```

StbMedia.GetZoom

```
<static> StbMedia.GetZoom(id)
```

Возвращает коэффициент зуммирования.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Коэффициент зуммирования.

Пример:

```
StbMedia.SetZoom(StbMedia.GetZoom(id) – 0.5);
```

StbMedia.IsMute

```
<static> StbMedia.IsMute(id)
```

Возвращает состояние режима "без звука".

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Состояние режима:

- 1 - без звука;
- 0 - со звуком.

StbMedia.IsPlayerVisible

```
<static> StbMedia.IsPlayerVisible(id)
```

Позволяет узнать отображается плеер или нет.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Состояние:

- 1 - плеер отображается;
- 0 - плеер не отображается.

Пример:

```
if(StbMedia.IsPlayerVisible(id) == 0)
  StbMedia.SetPlayerVisible(id, 1);
```

StbMedia.IsSubtitlesVisible

```
<static> StbMedia.IsSubtitlesVisible(id)
```

Позволяет узнать отображаются субтитры или нет.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Состояние субтитров:

- 1 - субтитры отображаются на экране;
- 0 - субтитры скрыты с экрана.

StbMedia.NextFrame

```
<static> StbMedia.NextFrame(id)
```

Переводит плеер на следующий кадр.
Функция работает только в режиме паузы.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Статус операции.

Пример:

```
function goNextFrame(){
 StbMedia.NextFrame(id);
}
StbMedia.onEvent = "onMediaEvent();";
function onMediaEvent() {
    var event = StbMedia.GetEventDescription();
    var event_type = parseInt(event.type);
    switch (event_type) {
    case ELT_PLAYER_STATUS_CHANGED:
               var state = StbMedia.GetPlayerState(id);
        if(state == PAUSED){
               setTimeout(goNextFrame, 3000);
        }
               break;
        ...
      }
}

var id = StbMedia.CreatePlayer();

... //запускаем на воспроизведение, ставим на паузу
```

StbMedia.Pause

```
<static> StbMedia.Pause(id)
```

Ставит плеер на паузу.
Когда воспроизведение будет приостановлено, произойдёт событие ELT_PLAYER_STATUS_CHANGED и статус плеера изменится на PAUSED.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Статус операции.

StbMedia.PlayFile

```
<static> StbMedia.PlayFile(id, path, pos, duration, fps, subt)
```

Запускает файл на воспроизведение.

Примечание:

При первом запуске файла происходит анализ мета-данных, который может занимать до нескольких секунд. Чтобы при повторном воспроизведении этого же файла избежать задержки, можно использовать FPS и duration, полученные при первом запуске. Когда плеер начнёт воспроизводить контент, произойдёт событие ELT_PLAYER_STATUS_CHANGED, статус плеера изменится на PLAYING_FWD_1x. Когда плеер распознает тип файла, произойдёт событие ELT_PLAYER_METADATA_UPDATED, далее можно запрашивать информацию о дорожках соответствующими вызовами API.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
path		путь к файлу "/tmp/video.avi"
pos		начальная позиция в файле в миллисекундах
duration	-1, если FPS = -1	продолжительность файла по времени в миллисекундах
fps	24, 25, 50, 60 и т.д.; ELT_AUTO_FPS = -1 - для автоопределения	количество кадров в секунду
subt		путь к файлу субтитров (.srt) или пустую строку

Возвращаемое значение:

Статус операции.

Пример:

```
var x = 100;
var y = 100;
var w = 300;
var h = 200;

StbMedia.SetPlayerViewPort(id, x,y,w,h);
StbMedia.PlayFile(id, "/tmp/video.avi", 0,-1,-1,"/tmp/sub.srt");
```

StbMedia.PlayHttp

```
<static> StbMedia.PlayHttp(id, path, opt)
```

Запускает на воспроизведение контент по HTTP-ссылке.

Примечание:

Когда плеер начнёт воспроизводить контент, произойдёт событие ELT_PLAYER_STATUS_CHANGED, статус плеера изменится на PLAYING_FWD_1x. Когда плеер распознает тип файла, произойдёт событие ELT_PLAYER_METADATA_UPDATED, далее можно запрашивать информацию о дорожках соответствующими вызовами API.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
path		HTTP-ссылка "http://videourl.com/somemedia.avi"
opt		специальный HTTP-заголовок, который будет использован плеером (передать NULL, если не нужно)

Возвращаемое значение:

Статус операции.

Пример:

```
var x = 500;
var y = 500;
var w = 300;
var h = 200;

StbMedia.SetPlayerViewPort(id, x,y,w,h);
StbMedia.PlayHttp(id, "http://videourl.com/somemedia.avi", NULL);
```

StbMedia.PlayMagnet

```
<static> StbMedia.PlayMagnet(id, magnet, name, size, pos, duration, fps)
```

Запускает файл на воспроизведение по magnet-ссылке.

Примечание:

Когда плеер начнёт воспроизводить контент, произойдёт событие ELT_PLAYER_STATUS_CHANGED, статус плеера изменится на PLAYING_FWD_1x. Когда плеер распознает тип файла, произойдёт событие ELT_PLAYER_METADATA_UPDATED, далее можно запрашивать информацию о дорожках соответствующими вызовами API.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
magnet		magnet-ссылка
name		имя файла
size		размер в килобитах
pos		позиция начала воспроизведения в миллисекундах
duration	-1, если FPS = -1	продолжительность файла по времени в миллисекундах
fps	24, 25, 50, 60 и т.д.; ELT_AUTO_FPS = -1 - для автоопределения	количество кадров в секунду

Возвращаемое значение:

Статус операции.

StbMedia.PlayRtsp

```
<static> StbMedia.PlayRtsp(id, path, mode)
```

Запускает проигрывание по RTSP-ссылке.

Примечание:

Когда плеер начнёт воспроизводить контент, произойдёт событие ELT_PLAYER_STATUS_CHANGED, статус плеера изменится на PLAYING_FWD_1x. Когда плеер распознает тип файла, произойдёт событие ELT_PLAYER_METADATA_UPDATED, далее можно запрашивать информацию о дорожках соответствующими вызовами API.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
path		ссылка "rtsp://videourl.com/somemedia.avi"
mode	raw, udp, rtp	тип транспортного потока

Возвращаемое значение:

Статус операции.

Пример:

```
var ELT_TRANSPORT_UDP = 1;
var ELT_TRANSPORT_RTP = 2;
var type = "udp";
...

var x = 500;
var y = 500;
```

```
var w = 300;
var h = 200;

StbMedia.SetPlayerViewPort(mp, x,y,w,h);

if(type != “udp”){
 StbMedia.PlayRtsp(id,"rtsp://videourl.com/somemedia.avi", 0, ELT_TRANSPORT_UDP);
}
else {
 StbMedia.PlayRtsp(id,"rtsp://videourl.com/somemedia.avi", 0, ELT_TRANSPORT_RTP);
}
```

StbMedia.PlayStream

<static> *StbMedia*.PlayStream(id, path)

Воспроизводит мультикастовый поток.

Примечание:

Чтобы уменьшить задержку переключения между потоками, можно не вызывать StbMedia.Stop(id), а вызывать StbMedia.PlayStream(id, path).

Когда плеер начнёт воспроизводить контент, произойдёт событие ELT_PLAYER_STATUS_CHANGED, статус плеера изменится на PLAYING_FWD_1x. Когда плеер распознает тип файла, произойдёт событие ELT_PLAYER_METADATA_UPDATED, далее можно запрашивать информацию о дорожках соответствующими вызовами API.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
path		URL поток "udp://226.0.0.1:1111" или "rtp://226.0.0.1:1111"

Возвращаемое значение:

Статус операции.

Пример:

```
    var x = 500;
    var y = 500;
    var w = 300;
    var h = 200;

    StbMedia.SetPlayerViewPort(id, x,y,w,h);
    StbMedia.PlayStream(id, "udp://226.0.0.1:1111");
```

StbMedia.SetAudioPid

```
<static> StbMedia.SetAudioPid(id, pid)
```

Выбирает аудиодорожку.
Когда дорожка переключилась, происходит событие ELT_PLAYER_AUDIO_TRACK_SWITCHED.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
pid		PID аудиодорожки

Возвращаемое значение:

Статус операции.

StbMedia.SetMute

```
<static> StbMedia.SetMute(id, mute)
```

Устанавливает режим "без звука".

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
mute	1 - без звука 0 - со звуком	состояние режима "без звука"

Возвращаемое значение:

Статус операции.

StbMedia.SetPlayerAlphaLevel

```
<static> StbMedia.SetPlayerAlphaLevel(id, alpha)
```

Устанавливает уровень прозрачности плеера.

Параметры:

Параметры	Значения	Описание
id	0 .. 255	идентификатор плеера
alpha	0 - полностью прозрачный	уровень прозрачности

Возвращаемое значение:

Статус операции.

StbMedia.SetPlayerDisplayMode

```
<static> StbMedia.SetPlayerDisplayMode(id, mode)
```

Устанавливает режим отображения видео.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
mode	DISPLAY_NORMAL = 1 — в нормальный DISPLAY_CROP = 2 — обрезанный по краям DISPLAY_STRETCH = 3 — растянутый	режим отображения

Возвращаемое значение:

Статус операции.

Пример:

```
var DISPLAY_NORMAL = 1;
var DISPLAY_CROP = 2;
var DISPLAY_STRETCH = 3;

StbMedia.SetPlayerDisplayMode(id, DISPLAY_CROP);
```

StbMedia.SetPlayerViewPort

```
<static> StbMedia.SetPlayerViewPort(id, x, y, w, h)
```

Вписывает плеер в прямоугольную область.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
x		координата x
y		координата y
w		ширина
h		высота

Возвращаемое значение:

Статус операции.

Пример:

```
if(!fullscreen)
 StbMedia.SetPlayerViewPort(id,500,500,300,200);
else
  StbDisplay.SetPlayerViewPort(id, 0,0, 1920, 1080);
```

StbMedia.SetPlayerVisible

```
<static> StbMedia.SetPlayerVisible(id, visibility)
```

Скрывает плеер с экрана, не останавливая воспроизведение, или возвращает плеер обратно на экран.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
visibility	1 - плеер отображается 0 - плеер не отображается	режим отображения

Возвращаемое значение:

Статус операции.

Пример:

```
if(StbMedia.IsPlayerVisible(id) == 0)
  StbMedia.SetPlayerVisible(id, 1);
```

StbMedia.SetPosition

```
<static> StbMedia.SetPosition(id, pos)
```

Установить позицию в миллисекундах.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
pos		позиция в миллисекундах, с которой необходимо начать/продолжить воспроизведение

Возвращаемое значение:

Статус операции.

Пример:

```
StbMedia.SetPosition(id, StbMedia.GetDuration(id)/2);
```

StbMedia.SetSpeed

<static> *StbMedia*.SetSpeed(id, speed)

Устанавливает скорость воспроизведения. Когда скорость увеличится, произойдёт событие ELT_PLAYER_STATUS_CHANGED и статус плеера изменится на PLAYING_FWD_xx или PLAYING_BWD_xx.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
speed		одна из доступных скоростей

Возвращаемое значение:

Статус операции.

Пример:

```
var ELT_SPEED_FWD_1X = 0x1;
var ELT_SPEED_FWD_2X = 0x2;
var ELT_SPEED_FWD_4X = 0x4;
var ELT_SPEED_FWD_8X = 0x8;
var ELT_SPEED_FWD_16X = 0x10;
var ELT_SPEED_FWD_32X = 0x20;
var ELT_SPEED_BWD_1X = 0x101;
var ELT_SPEED_BWD_2X = 0x102;
var ELT_SPEED_BWD_4X = 0x104;
var ELT_SPEED_BWD_8X = 0x108;
var ELT_SPEED_BWD_16X = 0x110;
var ELT_SPEED_BWD_32X = 0x120;
```

```
var speed = ELT_SPEED_FWD_1X;

function ffwd()
{
 switch(speed){
 case ELT_SPEED_FWD_1X:
        StbMedia.SetSpeed(id, ELT_SPEED_FWD_2X);
        break;
 case ELT_SPEED_FWD_2X:
        StbMedia.SetSpeed(id, ELT_SPEED_FWD_4X);
        break;
        default:
        StbMedia.SetSpeed(id, ELT_SPEED_FWD_1X);
 }
}

function fbwd()
{
 switch(speed){
 case ELT_SPEED_BWD_1X:
        StbMedia.SetSpeed(id, ELT_SPEED_BWD_2X);                    break;
 case ELT_SPEED_BWD_2X:
        StbMedia.SetSpeed(id, ELT_SPEED_BWD_4X);                    break;
 default:
        StbMedia.SetSpeed(id, ELT_SPEED_BWD_1X);
 }}
```

StbMedia.SetSubtitlePid

```
<static> StbMedia.SetSubtitlePid(id, pid)
```

Выбирает дорожку субтитров.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
pid		PID дорожки субтитров

Возвращаемое значение:

Статус операции.

StbMedia.SetSubtitlesVisible

```
<static> StbMedia.SetSubtitlesVisible(id, visibility)
```

Позволяет скрыть субтитры с экрана. При этом выбранная дорожка субтитров сохраняется.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
visibility	1 – показать субтитры 0 – скрыть субтитры	режим отображения

Возвращаемое значение:

Статус операции.

StbMedia.SetVolume

```
<static> StbMedia.SetVolume(id, vol)
```

Устанавливает уровень громкости.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
vol	0 - 72	уровень громкости

Возвращаемое значение:

Статус операции.

Пример:

```
var volume = 48;
...
function volumeUp(){
```

```
    if(volume < 72){
        volume++;
               StbMedia.SetVolume(id, volume);
    }
}

function volumeDown(){
    if(volume != 0){
        volume--;
               StbMedia.SetVolume(id, volume);
    }
}
```

StbMedia.SetZoom

```
<static> StbMedia.SetZoom(id, zoom)
```

Устанавливает коэффициент зуммирования, где 1 - без зуммирования.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера
zoom	1.0 - исходный размер; 0.5 - уменьшенный в два раза; 2.0 - увеличенный в два раза и т.д.	коэффициент зуммирования

Возвращаемое значение:

Статус операции.

Пример:

```
function zoomOut() {
 StbMedia.SetZoom(id,StbMedia.GetZoom(id) - 0.1);
}
function zoomIn() {
 StbMedia.SetZoom(id,StbMedia.GetZoom(id) + 0.1);
}
```

StbMedia.Stop

```
<static> StbMedia.Stop(id)
```

Останавливает воспроизведение. Когда плеер остановится, произойдёт событие ELT_PLAYER_STATUS_CHANGED и статус плеера изменится на STOPPED.

Параметры:

Параметры	Значения	Описание
id	0..255	идентификатор плеера

Возвращаемое значение:

Статус операции.

Приложение А Коды кнопок пульта дистанционного управления (ПДУ) в Java Script API

В таблице приведен список кнопок и соответствующие им коды, используемые в JavaScript API.

Таблица – Коды кнопок ПДУ

Кнопка ПДУ	Код кнопки, dec
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
0	58
ASTR	8
HASH	0
OK	13
RETURN	27
LEFT	37
RIGHT	39
DOWN	40
UP	38
HELP	47
PAUSE	19
INFO	115
RED	120
GREEN	121
YELLOW	122
BLUE	123
F1	112
F2	113
ONOFF	0; //кнопка не доступна
PLAY	400
NEXT	401
PREV	402
ZOOM	403
REC	404
KEYB	405
CAPS	406
VOLUP	407
VOLDN	408
AUDIO	409
SUBT	410
SEARCH	411
CHUP	412

CHDN	413
MUTE	414
MENU	415
STOP	416
REV	417
FWD	418
ROT	419

При нажатии на кнопку ПДУ формируется событие keyDown, вызывающее обработчик onKeyDown. Событие keyUp возникает сразу же после события keyDown, а не после отпускания кнопки пульта. Событие keyPress возникает только при нажатии на кнопки печатаемых символов, на управляющие кнопки событие keyPress не формируется.

```
<body onkeydown="keyHandler(event.keyCode);">
...
<script type="text/javascript">
...
var RC_RETURN_KEY = 27;
var RC_VOLUP_KEY = 407;
...
var RC_VOLDN_KEY = 408;
var RC_MUTE_KEY = 414;
var RC_STOP_KEY = 416;
...
function keyHandler(code){
        switch(code) {
                case RC_STOP_KEY:
                    stop();
                    break
                case RC_VOLUP_KEY:
                    volumeUp();
                    break
                case RC_VOLDN_KEY:
                    volumeDown();
                    break
                case RC_MUTE_KEY:
                    volumeMute();
                    break
                case RC_RETURN_KEY:
                    alert("return");
                    StbDisplay.ExitToMenu();
                    break
 }
}
...
</script>
</body>
```

Приложение В Примеры использования API

Пример 1. Воспроизведение мультикастового потока, изменение уровня громкости.

```
<!DOCTYPE html>

<html>
<head>

<style type="text/css" media="screen">

body {
    background-color:#FFEEAA;
    font-size:30px;
}
.player{
position: absolute;
background: #f0f0f0;
height: 480px;
width: 640px;
top: 50px;
left: 300px;
}

#volume{
    color: #BEBEBE;
    font: bold;
}

</style>
</head>

<body onunload="CloseAllPlayers();">

<div>Press OK to play</div>
<br>
<div id="volume">
<div>Current volume = <span id="currentVol"></span></div>
</div>

<script type="text/javascript">

var RC_OK_KEY = 13;
var RC_RETURN_KEY = 27;
var RC_VOLUP_KEY = 407;
var RC_VOLDN_KEY = 408;

var ELT_PLAYER_STATUS_CHANGED = 1;
var ELT_PLAYER_VOLUME_CHANGED = 7;

var volume = 48;
document.getElementById("currentVol").innerHTML = volume;
var playerActive = false;

function volumeUp(){
    if(volume < 72){
 volume++;
        StbMedia.SetVolume(mp, volume);
    }
}
```

```
function volumeDown(){
    if(volume != 0){
 volume--;
        StbMedia.SetVolume(mp, volume);
    }
}
function onMediaEvent(){

 var event = StbMedia.GetEventDescription();
 var event_type = parseInt(event.type);

 switch (event_type)
 {
        case ELT_PLAYER_STATUS_CHANGED:
               playerActive = true;
        break;
        case ELT_PLAYER_VOLUME_CHANGED:
               document.getElementById("currentVol").style.color = "#2F4F4F";
               document.getElementById("currentVol").innerHTML =
 StbMedia.GetVolume(mp);
        break;
 }
}
function keykey(code){
 var e = code.keyCode;
 switch(e){
        case RC_OK_KEY:
               StbMedia.PlayStream(mp, "udp://233.7.70.1:5000");
               break
        case RC_RETURN_KEY:
               alert("return");
                      StbDisplay.ExitToMenu();
                      break
        case RC_VOLUP_KEY:
               if(playerActive == true)
                      volumeUp();
                  break
               case RC_VOLDN_KEY:
               if(playerActive == true)
                             volumeDown();
                  break
 }
}

function CloseAllPlayers() {
 StbMedia.ClosePlayer(mp);
}
var mp = StbMedia.CreatePlayer();
var color = "f0f0f0";
StbDisplay.SetChromaKey(parseInt(color,16));
StbDisplay.SetChromaKeyEnable(1);
StbMedia.SetPlayerViewPort(mp, 300,50,640,480);
StbMedia.onEvent = "onMediaEvent();";

document.onkeydown=keykey;
</script>
<div class = "player"> </div>
</body>
</html>
```

Пример 2. Изменение уровня прозрачности пользовательского интерфейса.

```
<!DOCTYPE html>

<html>
<head>

<style type="text/css" media="screen">

body {
    background-color:#FFEEAA;
    font-size:30px;
}

</style>
</head>

<body>

<script type="text/javascript">

var RC_OK_KEY = 13;
var RC_RETURN_KEY = 27;

function keykey(code){
 var e = code.keyCode;
 switch(e){
        case RC_OK_KEY:
               alert("go!");
               alphalevel();
               break
        case RC_RETURN_KEY:
               alert("return");
                      StbDisplay.ExitToMenu();
                      break
 }
}

var alpha = 200;

StbDisplay.SetAlphaLevel(alpha);

function alphalevel(){
 while(alpha>0){
        alpha--;
        StbDisplay.SetAlphaLevel(alpha);
 }
 while(alpha<201){
        alpha++;
        StbDisplay.SetAlphaLevel(alpha);
 }
  }

document.onkeydown=keykey;

</script>
</body>
</html>
```

Пример 3. Вывод информации об устройстве.

```
<!DOCTYPE html>

<html>
<head>

<style type="text/css" media="screen">

body {
    background-color:#FFEEAA;
    font-size:30px;
}

</style>
</head>

<body>
<div id="background">
    <p>Device info</p>
    <div>JS API Version: <span id="APIVersion"> ss</span></div>
    <div>MAC address: <span id=macAddress></span></div>
    <div>IP address: <span id=ipAddress></span></div>
    <div>Firmware version: <span id=firmwareVersion></span></div>
    <div>Model: <span id=deviceModel></span></div>
    <div>User-Agent: <span id=serialNumber></span></div>
</div>

<script type="text/javascript">

var RC_RETURN_KEY = 27;

document.getElementById("APIVersion").innerHTML=StbDisplay.APIVersion;
document.getElementById("macAddress").innerHTML=StbDisplay.EthernetHardwareAddress;
document.getElementById("ipAddress").innerHTML=StbDisplay.IpAddress;
document.getElementById("firmwareVersion").innerHTML=StbDisplay.FirmwareVersion;
document.getElementById("deviceModel").innerHTML=StbDisplay.DeviceModel;
document.getElementById("serialNumber").innerHTML=StbDisplay.SerialNumber;

function keykey(code){
 var e = code.keyCode;
 switch(e){
        case RC_RETURN_KEY:
               alert("return");
                      StbDisplay.ExitToMenu();
                      break
 }
}

document.onkeydown=keykey;
</script>
</body>
</html>
```

Пример 4. Воспроизведение файла с флеш-накопителя, применение методов StbMedia.Stop(id), StbMedia.Pause(id), StbMedia.Continue(id), изменение скорости воспроизведения.

```
<!DOCTYPE html>

<html>
<head>

<style type="text/css" media="screen">

body {
    background-color:#FFEEAA;
    font-size:30px;
}
.player{
position: absolute;
background: #f0f0f0;
height: 480px;
width: 640px;
top: 50px;
left: 300px;
}

</style>
</head>

<body onunload="CloseAllPlayers();">
<div>Press OK to play</div>

<script type="text/javascript">

var RC_OK_KEY = 13;
var RC_PAUSE_KEY = 19;
var RC_RETURN_KEY = 27;
var RC_STOP_KEY = 416;
var RC_REW_KEY = 417;
var RC_FWD_KEY = 418;

var ELT_PLAYER_STATUS_CHANGED = 1;
var PAUSED = 2;

var ELT_SPEED_FWD_1X = 0x1;
var ELT_SPEED_FWD_1X = 0x1;
var ELT_SPEED_FWD_2X = 0x2;
var ELT_SPEED_FWD_4X = 0x4;
var ELT_SPEED_FWD_8X = 0x8;
var ELT_SPEED_FWD_16X = 0x10;
var ELT_SPEED_FWD_32X = 0x20;
var ELT_SPEED_BWD_1X = 0x101;
var ELT_SPEED_BWD_2X = 0x102;
var ELT_SPEED_BWD_4X = 0x104;
var ELT_SPEED_BWD_8X = 0x108;
var ELT_SPEED_BWD_16X = 0x110;
var ELT_SPEED_BWD_32X = 0x120;

var PLAYING_FWD_1x = 0x10000000 | ELT_SPEED_FWD_1X;

var playerActive = false;

function onMediaEvent(){
```

```
 var event = StbMedia.GetEventDescription();
 var event_type = parseInt(event.type);

 switch (event_type)
 {
 case ELT_PLAYER_STATUS_CHANGED:
        playerActive = true;
        speed = StbMedia.GetSpeed(event.id);
        alert("speed =" + speed);
 break;
 }
}

function stop(){
 alert("stop");
 StbMedia.Stop(mp);
}

function pause(){
 var state = StbMedia.GetPlayerState(mp);
        if(state == PAUSED){
               alert("continue");
               StbMedia.Continue(mp);
               playerActive = true;
        }
        else if(state & PLAYING_FWD_1x){
               alert("pause");
               StbMedia.Pause(mp);
               playerActive = false;
        }
}

var speed = ELT_SPEED_FWD_1X;

function ffwd()
{
 switch(speed){
 case ELT_SPEED_FWD_1X:
 StbMedia.SetSpeed(mp, ELT_SPEED_FWD_2X);
 break;
 case ELT_SPEED_FWD_2X:
 StbMedia.SetSpeed(mp, ELT_SPEED_FWD_4X);
 break;
 case ELT_SPEED_FWD_4X:
 StbMedia.SetSpeed(mp, ELT_SPEED_FWD_8X);
 break;
 case ELT_SPEED_FWD_8X:
 StbMedia.SetSpeed(mp, ELT_SPEED_FWD_16X);
 break;
 case ELT_SPEED_FWD_16X:
 StbMedia.SetSpeed(mp, ELT_SPEED_FWD_32X);
 break;
 case ELT_SPEED_FWD_32X:
 StbMedia.SetSpeed(mp, ELT_SPEED_FWD_1X);
 break;
 default:
 StbMedia.SetSpeed(mp, ELT_SPEED_FWD_1X);
 }
}

function fbwd()
```

```
{
 switch(speed){
 case ELT_SPEED_BWD_1X:
 StbMedia.SetSpeed(mp, ELT_SPEED_BWD_2X);
 break;
 case ELT_SPEED_BWD_2X:
 StbMedia.SetSpeed(mp, ELT_SPEED_BWD_4X);
 break;
 case ELT_SPEED_BWD_4X:
 StbMedia.SetSpeed(mp, ELT_SPEED_BWD_8X);
 break;
 case ELT_SPEED_BWD_8X:
 StbMedia.SetSpeed(mp, ELT_SPEED_BWD_16X);
 break;
 case ELT_SPEED_BWD_16X:
 StbMedia.SetSpeed(mp, ELT_SPEED_BWD_32X);
 break;
 case ELT_SPEED_BWD_32X:
 StbMedia.SetSpeed(mp, ELT_SPEED_BWD_1X);
 break;
 default:
 StbMedia.SetSpeed(mp, ELT_SPEED_BWD_1X);
 }
}

function keykey(code){
 var e = code.keyCode;
 switch(e){
        case RC_OK_KEY:
        StbMedia.PlayFile(mp, "/mnt/stb/local/usbDisk0/minitube/minitube.mp4", 0,-1,-
1,"/mnt/stb/local/usbDisk0/minitube/minitube.srt");
        break
        case RC_RETURN_KEY:
               alert("return");
                      StbDisplay.ExitToMenu();
                      break
        case RC_STOP_KEY:
               if(playerActive == true)
                     stop();
                  break
               case RC_PAUSE_KEY:
               if(playerActive == true)
                     pause();
                  break
        case RC_REW_KEY:
               if(playerActive == true)
                  fbwd();
                break
        case RC_FWD_KEY:
               if(playerActive == true)
                  ffwd();
               break
 }
}

function CloseAllPlayers() {
 StbMedia.ClosePlayer(mp);
}

var mp = StbMedia.CreatePlayer();

var color = "f0f0f0";
```

```
StbDisplay.SetChromaKey(parseInt(color,16));
StbDisplay.SetChromaKeyEnable(1);
StbMedia.SetPlayerViewPort(mp, 300,50,640,480);
StbMedia.onEvent = "onMediaEvent();";

document.onkeydown=keykey;
</script>
<div class = "player"> </div>
</body>
</html>
```

Пример 5. Выбор файла для воспроизведения из каталога.

```
<!DOCTYPE html>
<html>
<head>

<style type="text/css" media="screen">

body {
background-color:#FFEEAA;
font-size:30px;
}

.player{
position: absolute;
background: #f0f0f0;
height: 480px;
width: 640px;
top: 50px;
left: 500px;
}

#image{
position: absolute;
top: 250px;
left: 770px;
z-index: 1;
}

select {
width: 400px;
}
</style>
</head>

<body onunload="CloseAllPlayers();">

<div>'OK' on directory - play</div>
<div>'Right' on directory - next level of catalog</div>
<br>

<div id="listdir">
 <select size="10" id="mySelectId" name="mySelect" > </select>
</div>
<br>
<br>
<div>Current volume = <span id="currentVol"></span></div>

<div id="image">
 <img id="img" src="load.gif" width="100" height="100" style="display: none;"/>
```

```
</div>

<script type="text/javascript">

var RC_OK_KEY = 13;
var RC_PAUSE_KEY = 19;
var RC_RETURN_KEY = 27;
var RC_LEFT_KEY = 37;
var RC_UP_KEY = 38;
var RC_RIGHT_KEY = 39;
var RC_DOWN_KEY = 40;
var RC_VOLUP_KEY = 407;
var RC_VOLDN_KEY = 408;
var RC_MENU_KEY = 415;
var RC_STOP_KEY = 416;

var ELT_PLAYER_STATUS_CHANGED = 1;
var ELT_PLAYER_VOLUME_CHANGED = 7;

var LOADING = 1;
var PAUSED = 2;
var STOPPED = 3;

var ELT_SPEED_FWD_1X = 0x1;
var PLAYING_FWD_1x = 0x10000000 | ELT_SPEED_FWD_1X;

var volume = 48;
document.getElementById("currentVol").style.color = "#BEBEBE";
document.getElementById("currentVol").innerHTML = volume;

var playerActive = 0;

var pathStart = "/local/";
var pathCurrent = pathStart;
var pathPrev;

var objSel = document.getElementById("mySelectId");

function addOption (listbox, text, value, isDefaultSelected, isSelected){
 var opt = document.createElement("option");
 opt.appendChild(document.createTextNode(text));
 opt.setAttribute("value", value);
 if (isDefaultSelected) opt.defaultSelected = true;
 else if (isSelected) opt.selected = true;
 listbox.appendChild(opt);
}

function listDirs(path){
 var dir = StbDisplay.ListDirs(path);
 var files = StbDisplay.ListFiles(path);

 if(dir.status == "FAIL") {
        alert(fail);
        return;
 }
 if(dir.length != 0 || files.length != 0){
        objSel.options.length = 0;
        if(dir.length != 0){
               for(var i = 0; i<dir.length; i++)
                      addOption(objSel, dir[i]+"/", "dir", false, false);
        }
        if(files.length != 0){
```

```
                for(var i = 0; i<files.length; i++)
                        addOption(objSel, files[i], "file", false, false);
        }
 }
 else{
        pathCurrent = pathPrev;
 }
 objSel.options[2].selected = true;
}

function listUp(){
 if(objSel.selectedIndex > 0)
        objSel.selectedIndex--;
}

function listDown(){
 if(objSel.selectedIndex < objSel.options.length-1)
        objSel.selectedIndex++;
}

function listRight(){
 pathCurrent = pathCurrent + objSel.options[objSel.selectedIndex].text;
 if(objSel.options[objSel.selectedIndex].text == "local/")
 pathCurrent = "/local/";
 listDirs(pathCurrent);
 pathPrev = pathCurrent;
}

function onMediaEvent(){
 var event = StbMedia.GetEventDescription();
 var event_type = parseInt(event.type);

 switch (event_type)
 {
        case ELT_PLAYER_STATUS_CHANGED:
                playerActive = 1;
                playerStatusChanged();
        break;
        case ELT_PLAYER_VOLUME_CHANGED:
                document.getElementById("currentVol").style.color = "#2F4F4F";
                document.getElementById("currentVol").innerHTML =
StbMedia.GetVolume(mp);
        break;
 }
}

function playerStatusChanged(){
 var state = StbMedia.GetPlayerState(mp);
 switch(state){
        case LOADING:
                show();
        break;
        case STOPPED:
                playerActive = 0;
        break;
        default:
                hide();
        break;
 }
}

function volumeUp(){
```

```
    if(volume < 72){
        volume++;
        StbMedia.SetVolume(mp, volume);
    }
}

function volumeDown(){
    if(volume != 0){
        volume--;
        StbMedia.SetVolume(mp, volume);
    }
}

function stop(){
 alert("stop");
 StbMedia.Stop(mp);
 document.getElementById("currentVol").style.color = "#BEBEBE";
}

function pause(){
 var state = StbMedia.GetPlayerState(mp);
        if(state == PAUSED){
               alert("continue");
               StbMedia.Continue(mp);
               playerActive = 1;
        }
        else if(state & PLAYING_FWD_1x){
               alert("pause");
               StbMedia.Pause(mp);
               playerActive = 0;
        }
}

function show(){
 document.getElementById('img').style.display = 'block';
}

function hide(){
 document.getElementById('img').style.display = 'none';
}

function keykey(code){
 var e = code.keyCode;
 switch(e){
        case RC_OK_KEY:
               StbMedia.PlayFile(mp, pathCurrent+
objSel.options[objSel.selectedIndex].text, 0,-1,-1,"");
               break
        case RC_RETURN_KEY:
               alert("return");
               StbDisplay.ExitToMenu();
               break
        case RC_RIGHT_KEY:
               listRight();
               break
        case RC_UP_KEY:
               listUp();
               break
        case RC_DOWN_KEY:
               listDown();
               break
        case RC_VOLUP_KEY:
```

```
                if(playerActive == 1)
                        volumeUp();
                break
        case RC_VOLDN_KEY:
                if(playerActive == 1)
                        volumeDown();
               break
        case RC_STOP_KEY:
                if(playerActive == 1)
                        stop();
                break
        case RC_PAUSE_KEY:
                if(playerActive == 1)
                        pause();
                break
 }
}

function CloseAllPlayers() {
 StbMedia.ClosePlayer(mp);
}

listDirs(pathStart);
var mp = StbMedia.CreatePlayer();
var color = "f0f0f0";
StbDisplay.SetChromaKey(parseInt(color,16));
StbDisplay.SetChromaKeyEnable(1);
StbMedia.SetPlayerViewPort(mp, 500,50,640,480);
StbMedia.onEvent = "onMediaEvent();";

document.onkeydown=keykey;
</script>
<div class = "player"> </div>
</body>
</html>
```

ТЕХНИЧЕСКАЯ ПОДДЕРЖКА

Для получения технической консультации по вопросам эксплуатации оборудования ТОО «ЭлтексАлатау» Вы можете обратиться в Сервисный центр компании:

Республика Казахстан, 050032, г. Алматы, мкр. Алатау, ул. Ибрагимова, 9
Телефон:
+7(727) 320-18-38

E-mail: post@eltexalatau.kz

На официальном сайте компании Вы можете найти техническую документацию и программное обеспечение для продукции ТОО «ЭлтексАлатау» или проконсультироваться у инженеров Сервисного центра.
http://eltexalatau.kz