

---

# **Руководство по созданию пользовательских приложений для Samsung SMART TV**

---

Версия 1.20

Samsung Smart TV

Предисловие .....	7
Document History .....	7
1. Введение .....	8
1.1. Что такое Samsung Smart TV .....	8
2. Samsung Smart TV .....	8
2.1. Принцип действия .....	8
2.2. Рабочая среда.....	9
3. Приложения.....	10
3.1. Что собой представляет приложение Samsung Smart TV?.....	10
3.2. Структура приложения .....	10
3.3. Типы приложений .....	11
3.4. Запуск приложения .....	11
3.5. Менеджер приложений .....	12
3.6. Характеристики приложений Samsung.....	12
4. Создание приложений .....	13
4.1. Что необходимо для создания приложения.....	13
4.2. Как создать приложение.....	13
4.2.1. Создание файла config.xml.....	13
4.2.2. Создание файла index.html .....	14
4.2.3. Создание файла JavaScript .....	15
4.2.4. Создание файла CSS .....	16
4.2.5. Использование пульта дистанционного управления .....	16
4.2.6. Завершение создания тестового приложения .....	18
4.2.6.1. config.xml .....	18
4.2.6.2. index.html .....	19
4.2.6.3. Main.js .....	19
4.2.6.4. Main.css.....	20
4.3. config.xml .....	21
4.3.1. Описание тэгов .....	21
4.3.2. Пример.....	24
4.4. Аутентификация пользователей (Single-Sign-On) .....	25
4.4.1. Что такое система единого входа (Single-Sign-On) .....	25
4.4.2. Регистрация учетной записи службы провайдера .....	26
4.4.3. Как в приложении получить регистрационные данные службы провайдера.....	27

---

4.5. Переустановка (Reset) .....	28
4.5.1. Что такое переустановка .....	28
4.5.2. Как это работает.....	28
4.5.3. Что для этого должны реализовать разработчики .....	28
4.5.4. Инструкция по реализации config.xml .....	29
4.5.5. Инструкция по реализации модуля инициализации .....	29
4.6. Пример взаимодействия с объектом XMLHttpRequest .....	29
5. Общие модули.....	32
5.1. Общие модули Менеджера приложений.....	32
5.2. Как использовать общие модули .....	32
5.3. TVKeyValue .....	32
5.4. Widget.....	34
5.4.1. sendReadyEvent() .....	35
5.4.2. sendExitEvent() .....	35
5.4.3. sendReturnEvent() .....	35
5.4.4. blockNavigation().....	36
5.4.5. putInnerHTML() .....	37
5.4.6. getChannelWidgetListPath() .....	37
5.4.7. getSearchWidgetListPath() .....	38
5.4.8. runSearchWidget().....	38
5.4.9. checkSapTicket () .....	38
5.4.10. requestSapTicket ().....	39
5.5. Plugin.....	40
5.5.1. setOnWatchDog() .....	41
5.5.2. setOffWatchDog() .....	41
5.5.3. setOnOSDState() .....	42
5.5.4. setOffOSDState().....	42
5.5.5. registKey().....	42
5.5.6. unregistKey() .....	43
5.5.7. registIMEKey() .....	43
5.5.9. registAllKey.....	44
5.5.10. unregistAllKey() .....	44
5.5.11. registFullWidgetKey() .....	45
5.5.12. registPartWidgetKey().....	45

5.5.13. SetBannerState()	46
5.5.14. ShowTools()	46
5.5.15. setOnIdleEvent()	47
5.5.16. setOffIdleEvent()	47
5.5.17. setOnScreenSaver()	48
5.5.18. setOffScreenSaver()	48
5.5.19. isViewerKey()	49
5.5.20. setOnFullScreen()	49
5.5.21. setOffFullScreen()	49
5.6. CImageViewer	50
5.6.1. Что такое CImageViewer	50
5.6.2. Спецификации	50
5.6.3. Базовые инструкции	51
5.6.4. Анимационные эффекты при межкадровых переходах (Transition Effect)	51
5.6.4.1. Основные принципы создания слайд-шоу	52
5.6.4.2. Как реализовать анимационные эффекты	53
5.6.4.3. Слайд-шоу и ограничения	54
5.6.5. События	55
5.6.6. Воспроизведение локальных файлов	55
5.6.6.1. Воспроизведение изображений, хранящихся в приложении	55
5.6.6.2. Воспроизведение изображений с USB	56
5.6.7. Справка	57
5.6.7.1. CImageViewer	57
5.6.7.2. Effect	58
5.6.7.3. clearScreen()	58
5.6.7.4. destroy()	59
5.6.7.5. endSlideshow()	59
5.6.7.6. getStopFlag()	60
5.6.7.7. hide()	60
5.6.7.8. isEffectAvailable()	61
5.6.7.9. play()	61
5.6.7.10. prepareNext()	62
5.6.7.11. setFrameArea()	63
5.6.7.12. setOnBufferingComplete()	63

5.6.7.13. setOnBufferingStart()	64
5.6.7.14. setOnNetworkError()	64
5.6.7.15. setOnRenderError()	65
5.6.7.16. setOnRenderingComplete ()	65
5.6.7.17. show()	66
5.6.7.18. showNow ()	66
5.6.7.19. startSlideshow ()	67
5.6.7.20. stop ()	67
6. Рекомендации для разработчиков приложений	69
6.1. Функция, которая должна быть вызвана при открытии приложения.	69
6.2. Точка входа	69
6.3. Обработка событий, соответствующих нажатию кнопок на пульте управления	69
6.4. Кнопки пульта управления	70
6.5. Завершение приложения	71
6.6. Предотвращение закрытия приложения при нажатии кнопок Return или Exit.	71
6.7. Дополнительная информация	71
6.8. Что должен содержать CSS	72
6.9. Предварительная загрузка изображений	73
6.10. Особенности тэга <input>	73
6.11. Полоса прокрутки	73
6.12. Создание динамических веб-страниц.	73
6.13. Значение для межбуквенного расстояния	74
6.14. Управление памятью	74
6.14.1. Использование XMLHttpRequest	74
6.14.2. Освобождение памяти	74
6.15. Watch Dog	74
6.16. Предотвращение искажения экранных меню OSD (ON Screen Display)	74
6.17. Поддержка переполнения текста text-overflow в CSS	75
6.18. Инструкции по созданию видео-приложений.	75
6.19. Использование IME	75
6.20. Оптимизация скорости первоначальной загрузки приложения	75
6.20.1. Динамическая загрузка Javascript.	75
6.20.2. Динамическая загрузка фоновых изображений CSS	75
6.20.3. Разделение функций Main.onLoad	76

6.21. Инструкции по динамической загрузке стилей CSS .....	76
7. Браузер App Engine .....	78
7.1. Что собой представляет App Engine .....	78
7.2. Технические характеристики .....	78
8. Файловый программный интерфейс приложения (File API) .....	78
8.1. Что такое File API .....	78
8.2. Описание API .....	78
8.2.1. FileSystem() .....	79
8.2.2. openCommonFile() .....	79
8.2.3. closeCommonFile() .....	80
8.2.4. deleteCommonFile() .....	80
8.2.5. createCommonDir() .....	81
8.2.6. deleteCommonDir() .....	81
8.2.7. isValidCommonPath() .....	82
8.2.8. readLine() .....	82
8.2.9. writeLine() .....	83
8.2.10. readAll() .....	83
8.2.11. writeAll() .....	84
8.2.12. readDir() .....	84
8.2.13. openFile() .....	85
9. Справка .....	86
9.1. Код страны .....	86
9.2. Код языка .....	86
9.3. Код региона .....	87
9.4. Категория продукта .....	88
9.5. Предупреждение .....	88
9.5.1. Чувствительное к регистру устройство .....	88

## Предисловие

### Document History

Версия	Дата	Описание	Автор
v1.20	01/09/2011	Создание русской версии документа Application Development Guide for Samsung Smart TV - First Draft	Samsung Research Center, Moscow Smart TV Applications Support smarttv.ru@samsung.com

## 1. Введение

Этот раздел отвечает на вопросы, что собой представляет, из чего состоит и как работает Samsung Smart TV (Internet@TV).

### 1.1. Что такое Samsung Smart TV

Samsung Smart TV - это веб-приложение, которое выполняется браузером, встроенным в цифровой телевизор Samsung с доступом в интернет. Сервис Samsung Smart TV расширяет базовые функции телевизора, добавляя к ним набор разнообразных веб-функций, которые позволяют пользователям получать и просматривать интересную и полезную информацию на экранах своих телевизоров. Пользователи путем нехитрых операций могут получить доступ к различным интернет-сервисам, таким как новости, прогноз погоды, биржевые курсы и т.п.

С помощью Samsung Smart TV пользователи цифровых телевизоров могут не только скачивать и устанавливать приложения из каталога приложений HubSite, но также разрабатывать и устанавливать на телевизор свои собственные приложения.

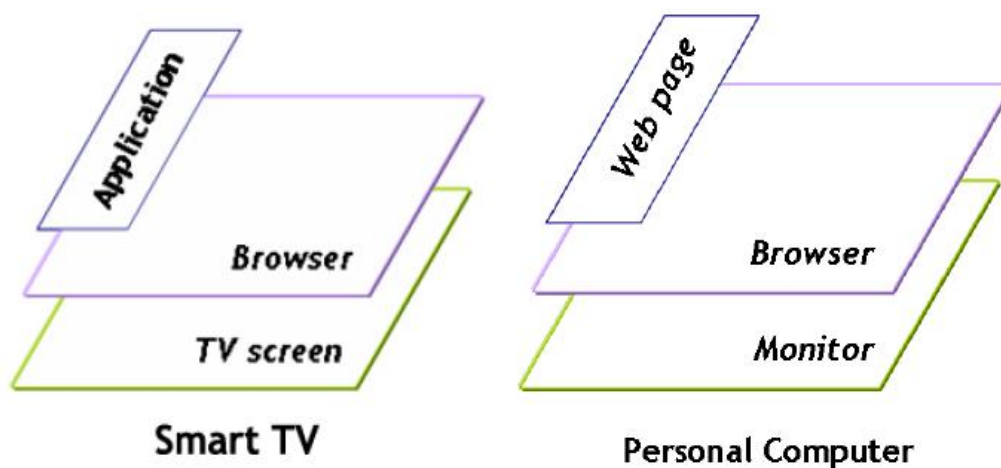


## 2. Samsung Smart TV

В этом разделе описаны принцип действия и рабочая среда Samsung Smart TV.

### 2.1. Принцип действия

Используя Samsung Smart TV, пользователи могут запускать на экране телевизора приложения, основанные на интернет-технологиях. Приложение (виджет) - это веб-страница определенного типа, которая выполняется веб-браузером на экране телевизора. Просмотр такой страницы аналогичен просмотру веб-страниц на обычном компьютере. Отличия заключаются в разрешении экрана, аппаратном обеспечении и в интерфейсе пользователя, который представляет собой пульт дистанционного управления.



## 2.2. Рабочая среда

<b>Браузер</b>	App Engine 6.0
<b>Разрешение экрана</b>	1280x720 pixel, 960x 540 pixel

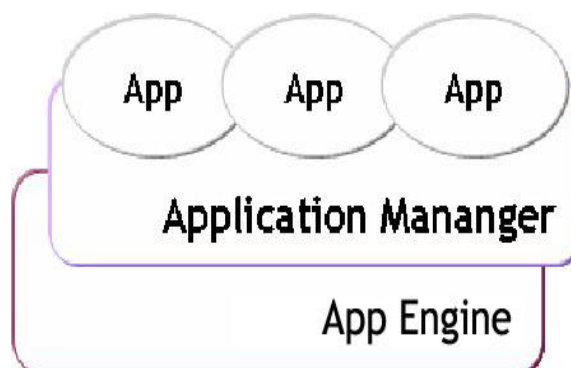
Подробное описание браузера App Engine (Widget Engine) для телевизоров Samsung приводится в разделе 7.

## 3. Приложения

Этот раздел содержит краткий обзор приложений, которые мы будем разрабатывать для телевизоров Samsung.

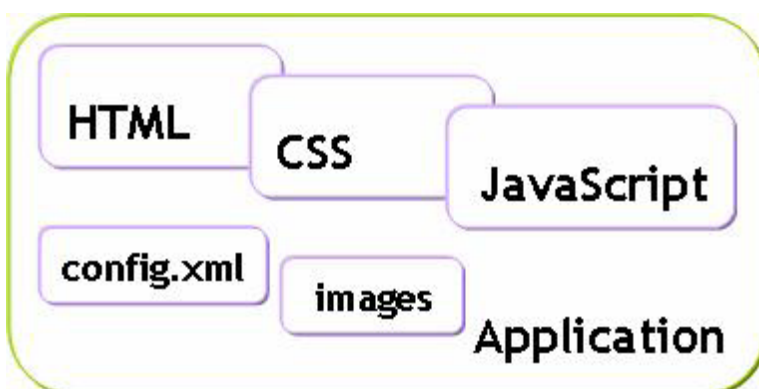
### 3.1. Что собой представляет приложение Samsung Smart TV?

Приложения Samsung Smart TV - это небольшие веб-приложения, которые выполняются на экране цифрового телевизора, имеющего доступ в интернет. Они устанавливаются на телевизор через интернет и выполняются в браузере App Engine. В общем случае можно сказать, что приложение - это веб-страница, которую пользователь просматривает на экране телевизора, а в качестве единственного пользовательского интерфейса выступает пульт дистанционного управления. Система управления приложениями Application Manager, которую в дальнейшем будем называть Менеджером приложений (Smart Hub, N.Navi.), осуществляет установку, удаление приложений и ряд других операций. Для разработки новых приложений для вашего телевизора вы можете использовать предназначенный для этого инструментарий SDK.






### 3.2. Структура приложения

Как уже упоминалось, приложение - это веб-страница, состоящая из HTML, CSS и JavaScript, которая выполняется в браузере App Engine. HTML страница описывает структуру приложения, CSS файл отвечает за стиль, а JavaScript файл управляет логикой приложения. Если вы хотите, чтобы созданное вами приложение выполнялось на экране телевизора, вы должны создать файл config.xml, содержащий информацию об оперативной среде и версии приложения. Подробнее о файле config.xml читайте в разделе 4.3.



### 3.3. Типы приложений

Тип	Описание	Расположение на экране
Полноэкранное приложение	Занимает все пространство экрана	
Оконное приложение (Single-wide)	Занимает часть экрана	
Бегущая строка (Ticker)	Приложение остается на экране, в то время как пользователь выполняет другие действия со своим телевизором	

**\* Для Европы доступны только полноэкранные приложения**

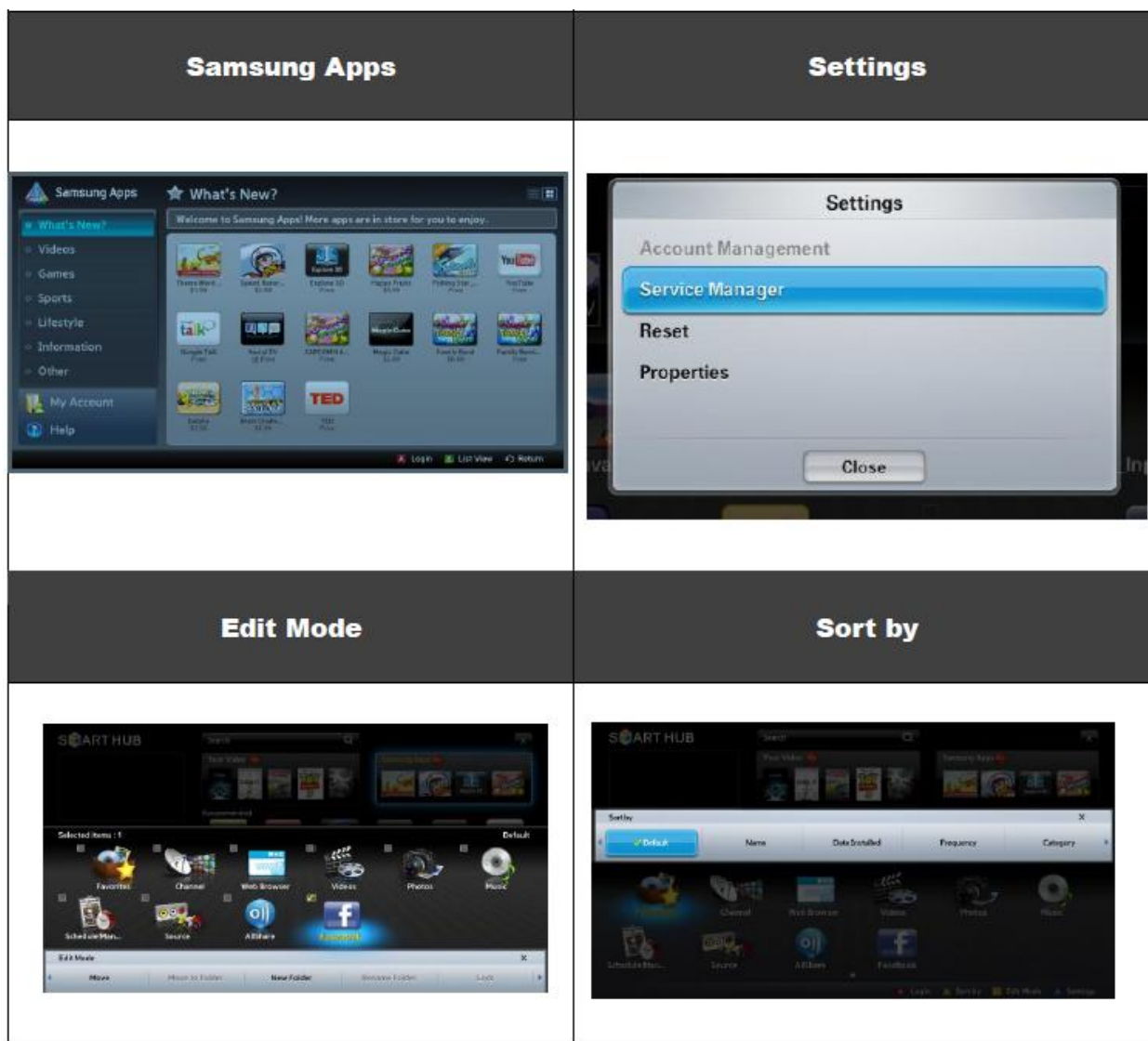
### 3.4. Запуск приложения

Чтобы запустить приложение, вы должны выполнить следующие шаги:

Шаг	Описание действия
Запустите Менеджер приложений	Нажмите кнопку INTERNET(INFO.L, SMART HUB) на пульте управления телевизором или кнопку Menu и далее последовательно Application>Samsung Smart TV
Выберите приложение	С помощью стрелок на пульте управления выберите интересующее вас приложение
Запустите приложение	Нажмите кнопку Enter для запуска приложения. Менеджер приложений читает информацию из файла config.xml запущенного приложения для того, чтобы установить его рабочие параметры, и затем читает файл index.html для того, чтобы запустить приложение.

### 3.5. Менеджер приложений

Менеджер приложений отвечает за выполнение функций, таких как аутентификация, установка, удаление, обновление, открытие и завершение приложений. С помощью Менеджера приложений пользователи могут устанавливать или удалять приложения, а также регистрировать свои собственные приложения и запускать их на своем телевизоре. Менеджер приложений - это тоже приложение, которое постоянно обновляется через интернет.



### 3.6. Характеристики приложений Samsung

Отличие приложений Samsung Smart TV от обычных веб-приложений заключается еще и в том, что здесь вы имеете возможность пользоваться функциями, которые есть только в телевизоре. Например, используя плагины, в приложении вы можете управлять громкостью звука и проигрывать видео. Кроме того, вы можете использовать файловую систему. Если вы хотите получить более подробную информацию о плагинах и файловой системе, смотрите разделы 7 и 8. Менеджер приложений поддерживает несколько JavaScript модулей (библиотек), использование которых значительно облегчает создание приложений. Для получения более детальной информации по использованию общих модулей, см. раздел 5.

## 4. Создание приложений

Этот раздел содержит инструкции по созданию простейшего приложения.

### 4.1. Что необходимо для создания приложения

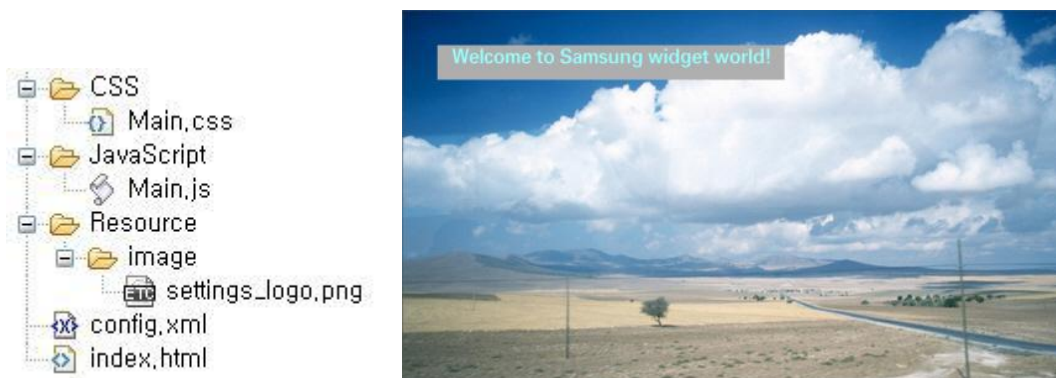
Нашей целью является создание приложения и запуск его на экране телевизора. Для этого вам понадобится телевизор Samsung с доступом в интернет и средство разработки SDK или текстовый редактор для написания HTML, JavaScript и CSS файлов. SDK снабжен эмулятором, который позволяет протестировать созданное вами приложение перед тем, как оно будет установлено на телевизор. Таким образом, использование SDK может сделать процесс создания приложений более удобным.

### 4.2. Как создать приложение

Приложение должно содержать, как минимум, следующие элементы:

- Файл index.html. Этот файл служит точкой входа в приложение.
- Конфигурационный файл приложения config.xml. Это XML файл в корневом каталоге приложения, который содержит информацию об установочных настройках приложения.
- Файлы JavaScript. Используются для управления логикой приложения.
- Файлы CSS. Эти файлы определяют внешний вид приложения.
- Файлы изображений (картинки), которые использует ваше приложение.

Давайте создадим приложение, которое выводит на экран текст и реагирует на нажатие кнопок на пульте управления. Структура и внешний вид приложения, которое мы будем сейчас создавать, приведены ниже.



#### 4.2.1. Создание файла config.xml

Файл config.xml вызывается первым из всех файлов, которые содержит приложение. Значение тэга `<ver>` определяет, надо ли обновлять приложение, а изображение, указанное в тэге `<ThumbIcon>` будет использоваться в качестве пиктограммы для приложения. Более детальную информацию о тэгах в файле config.xml см. в разделе 4.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<widget>
  <ThumbIcon>Resource/image/icon/picasa_106.png</ThumbIcon>
  <BigThumbIcon>Resource/image/icon/picasa_115.png</BigThumbIcon>
```

```

<ListIcon>Resource/image/icon/picasa_85.png</ListIcon>
<BigListIcon>Resource/image/icon/picasa_95.png</BigListIcon>

<category>lifestyle</category>
<autoUpdate>y</autoUpdate>
<cpname>MyCP</cpname>
<cpauthjs></cpauthjs>
<login>y</login>

<ver>0.930</ver>
<mgrver>1.000</mgrver>

<fullwidget>n</fullwidget>
<srcctl>n</srcctl>
<ticker>n</ticker>
<childlock>n</childlock>
<audiomute>n</audiomute>
<videomute>n</videomute>
<dcont>y</dcont>
<network>y</network>
<hubsite>n</hubsite>

<widgetname>HelloWorld</widgetname>
<description>Welcome!</description>

<width>960</width>
<height>540</height>

<author>
  <name>Samsung Electronics Co. Ltd.</name>
  <email></email>
  <link>http://www.sec.co.kr/</link>
  <organization>Samsung Electronics Co. Ltd.</organization>
</author>
</widget>

```

#### 4.2.2. Создание файла index.html

Следующим шагом будет создание файла index.html, который является точкой входа в приложение. Приведенный ниже пример HTML кода включает ссылку на файл Main.js, содержащийся в папке JavaScript, функция Main.onLoad() которого вызывается в момент загрузки документа.

```

<!DOCTYPE html>

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Hello World!!</title>
  <script type="text/javascript" language="javascript" src="JavaScript/Main.js"></script>
</head>

<body onload="Main.onLoad();">
  <div>Welcome to Samsung application world!</div>
</body>
</html>

```

### 4.2.3. Создание файла JavaScript

В момент загрузки HTML документа, вызывается функция `onLoad()` объекта `Main`, потому что вы указали функцию `Main.onLoad()` в атрибуте `onload` тэга `<body>`. Создайте объект `Main` и добавьте функцию `onload`.

```
var Main = {                                     // объект Main
}

Main.onLoad = function () {                     // вызывается по событию onload тэга <body>
    alert("Main.onLoad()");

    /**
     * JavaScript код Здесь!
     */
}
```

Если вы проделали все эти действия и попытались запустить созданное приложение в эмуляторе SDK, вы могли увидеть отладочное сообщение `"Main.onLoad()"`, но ожидаемая надпись „Welcome to Samsung application world!“ на экране телевизора не появилась. Если приложение успешно загружено, чтобы быть показанным на экране, оно должно вызвать специальную функцию т.н. общего модуля, который поддерживает Менеджер приложений. Общий модуль - это библиотека, содержащая специальные функции Менеджера приложений. Для получения более детальной информации см. раздел 5 Общие Модули.

Добавьте следующий код к тэгу `<head>` файла `index.html`.

```
<script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/Widget.js"></script>
```

Если вышеприведенный код добавлен, общий модуль становится доступным для использования в вашем JavaScript файле. В файле `Main.js` объявите общий модуль как глобальную переменную, которую вы будете использовать, и вызовите функцию `sendReadyEvent()`. Таким образом, вы даете команду Менеджеру приложений отобразить приложение на экране.

```
var Main = {                                     // Объект Main
}

var widgetAPI = new Common.API.Widget(); // Создание экземпляра общего модуля

Main.onLoad = function () {                     // вызывается по событию onload тэга <body>
    alert("Main.onLoad()");
    widgetAPI.sendReadyEvent();                 // Сообщение Менеджеру приложений о готовности
    /**
     * JavaScript код Здесь!
     */
}
```

Запустите созданное вами приложение. Теперь предложение “Welcome to Samsung application world!”, которое вы ввели в файле index.html, появится на экране. Возможно, вам захочется улучшить внешний вид вашего приложения, т.к. размер шрифта слишком маленький, а его цвет всего лишь черный. Для того чтобы сделать ваше приложение более стильным, вы должны использовать в нем стили CSS.

#### 4.2.4. Создание файла CSS

Добавьте следующую строку к тэгу <head> файла index.html.

```
<link rel='stylesheet' type='text/css' href = 'CSS/Main.css'/>
```

Присвойте ID тэгу <div> в файле index.html.

```
<div id='welcome'>Welcome to Samsung application world!</div>
```

Создайте файл в папке CSS и введите в него код, приведенный ниже, который описывает стиль элемента “welcome”.

```
body {
    margin: 0;
    padding: 0;
    background-color: transparent;
}

#welcome {
    position: absolute;
    left: 50px;
    top: 50px;
    width: 500px;
    height: 50px;

    background-color: #AFAFAF;
    color: #99FFFF;
    font-size: 30px;
    text-align: center;
}
```

#### 4.2.5. Использование пульта дистанционного управления

Сейчас мы изменим приложение так, чтобы оно могло реагировать на нажатие кнопок на пульте дистанционного управления. С помощью нажатия одной из пяти кнопок, находящихся в центре пульта, вы сможете изменять текст сообщения, которое выводится на экран.

При нажатии кнопки на пульте управления происходит событие “keydown”. В файле index.html должен быть описан элемент, который будет реагировать на это событие. Добавьте элемент <a> и в свойстве onkeydown укажите функцию, которая будет обрабатывать событие нажатия кнопки. Для того чтобы эта функция была выполнена, на экране телевизора нужно будет созданный элемент сделать активным (поместить на него фокус) и нажать кнопку на пульте управления.

В файле index.html добавляем тэг <a> с функцией Main.keyDown(), которая будет обрабатывать событие „keydown“.

```
<body onload="Main.onLoad();">
  <div id='welcome'>Welcome to Samsung application world!</div>
  <a href='javascript:void(0);' id='anchor' onkeydown='Main.keyDown();'></a>
</body>
```

В файле Main.js создаем функцию keyDown(), в которой получаем код кнопки, нажатой на пульте управления.

```
Main.keyDown = function(){                                // Обработчик нажатия кнопки
    var keyCode = event.keyCode;
    alert("Main Key code : " + keyCode);
}
```

В функции-обработчике, такой как keydown(), каждая кнопка имеет свой собственный код. Общий модуль “TVKeyValue” Менеджера приложений содержит значения кодов для кнопок пульта дистанционного управления.

Чтобы иметь возможность работать с общим модулем “TVKeyValue”, добавьте следующую строку к тэгу <head> в файле index.html.

```
<script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/TVKeyValue.js"></script>
```

Изменим код файла Main.js так, чтобы содержимое тэга div „welcome” менялось при нажатии кнопки. Сначала необходимо создать объекты общих модулей, в функции keydown() получить значение кода нажатой кнопки и для каждого из возможных значений прописать необходимые действия.

Для получения более детальной информации об общих модулях (библиотеках) и списка значений кодов кнопок см. Раздел 5 Общие Модули.

```
var Main = {                                              // Объект Main
}

var widgetAPI = new Common.API.Widget();                // Создание объекта общего модуля
var tvKey = new Common.API.TVKeyValue();

Main.onLoad = function(){                                // вызывается по событию onload тэга <body>

    alert("Main.onLoad()");
    widgetAPI.sendReadyEvent();                          // Сообщение Менеджеру приложений о готовности
    document.getElementById("anchor").focus();          // Помещение фокуса на элемент

    /**
     * JavaScript код Здесь!
     */
}
```

```

Main.keyDown = function(){                                     // Обработчик нажатия кнопки
    var keyCode = event.keyCode;
    alert("Main Key code : " + keyCode);

    switch (keyCode) {
        case tvKey.KEY_LEFT:
            alert("left");
            document.getElementById("welcome").innerHTML = "Nice to meet you.";
            /**
             * Code for Left key event!
             */
            break;
        case tvKey.KEY_RIGHT:
            alert("right");
            document.getElementById("welcome").innerHTML = "I'm so happy.";
            break;
        case tvKey.KEY_UP:
            alert("up");
            document.getElementById("welcome").innerHTML = "I Love you.";
            break;
        case tvKey.KEY_DOWN:
            alert("down");
            document.getElementById("welcome").innerHTML = "Good job.";
            break;
        case tvKey.KEY_ENTER:
            alert("enter");
            break;
        case tvKey.KEY_RETURN:
            break;
    }
}

```

Теперь вы можете видеть, что на экране содержимое тэга div „welcome“ будет изменяться при нажатии кнопок вверх, вниз, влево или вправо.

#### 4.2.6. Завершение создания тестового приложения

В приложениях Samsung Smart TV вы можете делать почти всё то же самое, что и на обычной веб-странице. Используя плагины, вы также можете создавать TV-ориентированные приложения с разнообразной функциональностью.

Ниже приведен код всех файлов приложения, которое мы создали в этом разделе.

##### 4.2.6.1. config.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<widget>
    <ThumbIcon>Resource/image/icon/picasa_106.png</ThumbIcon>
    <BigThumbIcon>Resource/image/icon/picasa_115.png</BigThumbIcon>

    <ListIcon>Resource/image/icon/picasa_85.png</ListIcon>
    <BigListIcon>Resource/image/icon/picasa_95.png</BigListIcon>
    <previewjs>PreviewHelloWorld</previewjs>
    <cname>Samsung</cname>
    <cplogo>Resource/image/settings_logo.png</cplogo>
    <cpauthjs></cpauthjs>
    <ver>0.930</ver>

```

```

<mgrver>1.000</mgrver>

<fullwidget>n</fullwidget>
<srcctl>n</srcctl>
<ticker>n</ticker>
<childlock>n</childlock>
<audiomute>n</audiomute>
<videomute>n</videomute>
<dcont>y</dcont>
<network>y</network>
<hubsite>n</hubsite>

<widgetname>HelloWorld</widgetname>
<description>Welcome!</description>

<width>960</width>
<height>540</height>
<author>
    <name>Samsung</name>
    <email></email>
    <link>http://acme-widget.example.com</link>
    <organization>Acme Examples, Inc.</organization>
</author>
</widget>

```

#### 4.2.6.2. index.html

```

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Hello World!</title>
    <script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/Widget.js"></script>
    <script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/TVKeyValue.js"></script>
    <script type="text/javascript" language="javascript" src="JavaScript/Main.js"></script>

    <link rel='stylesheet' type='text/css' href = 'CSS/Main.css'/>
</head>

<body onload="Main.onLoad();">
    <div id='welcome'>Welcome to Samsung widget world!</div>
    <a href='javascript:void(0);' id='anchor' onkeydown='Main.keyDown();'></a>
</body>
</html>

```

#### 4.2.6.3. Main.js

```

var Main = {                                // Объект Main
}

var widgetAPI = new Common.API.Widget();    // Создание объекта общего модуля
var tvKey = new Common.API.TVKeyValue();
Main.onLoad = function() {                  // вызывается по событию onload тэга <body>

```

```

alert("Main.onLoad()");
widgetAPI.sendReadyEvent(); // Сообщение Менеджеру приложений о готовности
document.getElementById("anchor").focus(); // Помещение фокуса на элемент

/**
 * JavaScript code Here!
 */
}

Main.keyDown = function(){ // Обработчик нажатия кнопки
    var keyCode = event.keyCode;
    alert("Main Key code : " + keyCode);

    switch (keyCode) {
        case tvKey.KEY_LEFT:
            alert("left");
            document.getElementById("welcome").innerHTML = "Nice to meet you.";
            /**
             * Code for Left key event!
             */
            break;
        case tvKey.KEY_RIGHT:
            alert("right");
            document.getElementById("welcome").innerHTML = "I'm so happy.";
            break;
        case tvKey.KEY_UP:
            alert("up");
            document.getElementById("welcome").innerHTML = "I Love you.";
            break;
        case tvKey.KEY_DOWN:
            alert("down");
            document.getElementById("welcome").innerHTML = "Good job.";
            break;
        case tvKey.KEY_ENTER:
            alert("enter");
            break;
        case tvKey.KEY_RETURN:
            break;
    }
}

```

#### 4.2.6.4. Main.css

```

body {
    margin: 0;
    padding: 0;

    background-color: transparent;
}

#welcome {
    position: absolute;
    left: 50px;
    top: 50px;
    width: 500px;
    height: 50px;
}

```

```
background-color: #AFAFAF;
color: #99FFFF;
font-size: 30px;
text-align: center;
}
```

### 4.3. config.xml

Файл config.xml содержит информацию, касающуюся выполнения, обновления, настроек приложения, и другие связанные с этим параметры. В соответствии с этой информацией Менеджер приложений контролирует версию приложения, устанавливает его конфигурационные параметры, создает и управляет учетными записями пользователя. Файл config.xml должен находиться в той же директории, в которой находится установленное приложение, и может содержать следующие элементы.

#### 4.3.1. Описание тэгов

Элемент	Описание	Значение
<widget>	Внешний тэг, определяющий принадлежность информации к приложению	-
<ThumbIcon>	Пиктограмма, которая отображается в меню Менеджера приложений (неактивная). Размер 106 x 86 пикселей.	Путь к файлу
<BigThumbIcon>	Пиктограмма, которая отображается в меню Менеджера приложений (активная). Размер 115 x 95 пикселей.	Путь к файлу
<ListIcon>	Пиктограмма, которая отображается в меню Менеджера приложений (список). Размер 85 x 70 пикселей.	Путь к файлу
<BigListIcon>	Пиктограмма, которая отображается в меню Менеджера приложений (активная в списке). Размер 95 x 78 пикселей.	Путь к файлу
<category>	Категория приложения. В списке доступных категорий определены video, sports, game, lifestyle, information (видео, спорт, игры, информация, стиль жизни) и другие.	Строка
<autoUpdate>	Определяет, синхронизовано ли приложение со службой hub site (с каталогом приложений). Для приложений, не нуждающихся в синхронизации, установлено значение "n".	у   n
<apptype>	Тип приложения - 11: HTML + java-script + Flash Player Object - 12: Adobe SWF ( Ver. Flash Lite 3.1 ) - 13: Adobe SWF ( Ver. Flash 10.1 ) - 14 : Lua Script	Число

<contents>	Путь к файлу, выполняющему инициализацию контента. Необходим только для приложений с следующими значениями apptype. - 12: Adobe SWF ( Ver. Flash Lite 3.1 ) - 13: Adobe SWF ( Ver. Flash 10.1 ) - 14: Lua Script	Путь к файлу
<channelType>	Channel Bound Service Type (опционально)	root   child
<channelRoot>	Confirms the relations with root-child clarifying root application ID. (Optional, Only when the channel bound service type is child.) When connected to more than one root, roots are distinguished by ::.	ID
<channelName>	Channel information to be executed channel bound service (Для разделения каналов используется знак :: Пример AAA::BBB::CCC) (Опционально, только если в качестве channel bound service type выбран root.)	Строка
<channelDisplay>	It decides whether the installed channelbound service is displayed on the first main screen or not. (If you want to hide the service in the first main screen, select „n”.)	y   n
<previewjs>	<i>устарел</i>	Строка
<cpname>	Имя контент-провайдера.	Строка
<cpauthjs>	Имя JavaScript файла, который выполняет проверку учетной информации служб провайдеров. Этот файл должен быть написан в соответствии с установленным форматом.	Строка
<cplogo>	<i>устарел</i>	Путь к файлу
<prelcon>	<i>устарел</i>	Путь к файлу
<login>	Определяет, нужна ли авторизация для доступа к сайту (службе) провайдера. Если выбрано значение “у”, то вы можете зарегистрировать логин и пароль через встроенную службу SSO Менеджера приложений. Проверка полномочий должна осуществляться в файле, имя которого указано в тэге <cpauthjs>.	y   n
<ver>	Версия приложения. В соответствии с указанным значением версии, сервер обновляет соответствующее приложение.	x.xxx
<mgrver>	Версия Менеджера приложений (версия Smart Hub), которая требуется для запуска приложения.	x.xxx
<fullwidget>	Определяет, является ли приложение полноэкранным или узкоформатным. Это влияет на аудио воспроизведение во время выполнения приложения. Для получения более подробной информации см. раздел 6.4. Использование кнопок пульта управления в приложениях	y   n

<srcctl>	Если необходимо переключение источника сигнала, установите этот параметр равным “у” (исключение - приложение Youtube). Если вы выбираете значение “у”, то источник сигнала автоматически переключается с текущего телевизионного канала или внешнего входа на внутренний медиа плеер и обратно, когда приложение завершается.	y   n
<childlock>	Определяет, доступна ли в приложении функция childlock, которая позволяет пользователю блокировать приложение.	y   n
<audiomute>	Включить/выключить аудио. Если выбрано значение “у”, то звук телевизионной трансляции не выводится в момент выполнения приложения. Установите “у” для полноэкранного приложения и “н”, если приложение занимает только часть экрана.	y   n
<videomute>	Включить/выключить видео. Если установлено значение “у”, телевизионная трансляция в момент выполнения приложения не выводится на экран.	y   n
<dcont>	Отключить функцию динамической настройки контраста. Динамическая настройка контраста - это функция, которая во время трансляции регулирует контрастность и яркость экрана телевизора, затемняя темные участки и осветляя светлые. Поэтому, если она включена, в момент выполнения приложения экран может периодически становиться то ярче, то темнее. Для отключения функции устанавливается значение “у”. Выберите “у” для полноэкранного приложения и “н”, если приложение занимает только часть экрана.	y   n
<movie>	Приложение, проигрывающее видео-файлы, может столкнуться со следующими проблемами. 1. Наложение звуков в момент выполнения приложения при преобразовании источников, если видео-файл проигрывается на устройстве, подсоединенном по HDMI порту, например на DVD-плеере. 2. Мерцание изображения в момент выполнения приложения из-за разницы частоты смены кадров между телевизионной картинкой и проигрыванием видео-файла. Если вы установите значение “у”, вы сможете предотвратить эти проблемы путем завершения работы подсоединенного по HDMI порту устройства и регулировки частоты смены кадров.	y   n
<widgetname>	Наименование приложения.	Строка
<description>	Краткое описание приложения.	Строка
<width> <height>	Область экрана, которую будет занимать приложение. В большинстве случаев, рекомендуется вводить значения 960 * 540 в соответствии со спецификацией цифрового телевизора.	Число
<author>	Информация об авторе приложения.	Строка
<network>	Этот тэг задает необходимость проверки сетевого соединения во время выполнения приложения. Если установлено значение “у”, а результат проверки сети - “сбой”, то вход в приложение будет заблокирован с выводом сообщения об ошибке. По умолчанию установлено значение “у”.	y   n

<hubsite>	Используется, если в момент выполнения приложения необходимо проверять, авторизована ли служба hubsite или нет. Если установлено значение “у”, в случае, когда hubsite не авторизован, вход в приложении блокируется с выводом соответствующего сообщения об ошибке. По умолчанию установлено значение “н”.	y   n
<pushNotice>	Определяет, доступен ли в приложении сервис рассылки уведомлений. По умолчанию установлено значение “н”.	y   n
<pushControl>	Этот тэг зарезервирован под шаблон уведомлений. По умолчанию установлено значение “н”.	y   n
<pushUerbinding>	Определяет, доступен ли в приложении сервис рассылки уведомлений для конкретного пользователя. По умолчанию установлено значение “н”.	y   n
<flashplayer>	Этот тэг указывается для приложений, которые используют встроенный или автономный flash плеер	y   n

#### 4.3.2. Пример

```

<?xml version="1.0" encoding="UTF-8"?>
<widget>
  <ThumbIcon>Resource/image/icon/picasa_106.png</ThumbIcon>
  <BigThumbIcon>Resource/image/icon/picasa_115.png</BigThumbIcon>
  <ListIcon>Resource/image/icon/picasa_85.png</ListIcon>
  <BigListIcon>Resource/image/icon/picasa_95.png</BigListIcon>

  <category>lifestyle</category>
  <autoUpdate>y</autoUpdate>

  <cpname>Picasa</cpname>
  <cpauthjs>Auth11101000000</cpauthjs>
  <login>y</login>
  <ver>2.000</ver>
  <mgrver>1.035</mgrver>
  <fullwidget>y</fullwidget>
  <srcctl>n</srcctl>
  <ticker>n</ticker>
  <childlock>n</childlock>
  <audiomute>y</audiomute>
  <videomute>y</videomute>
  <dcont>y</dcont>
  <movie>y</movie>
  <network>y</network>
  <hubsite>n</hubsite>
  <widgetname>Picasa Web Albums</widgetname>
  <description>View photo albums online</description>
  <width>960</width>
  <height>540</height>
  <author>
    <name>Samsung Electronics Co. Ltd.</name>
    <email></email>
    <link>http://www.sec.co.kr</link>
    <organization>Samsung Electronics Co. Ltd.</organization>
  </author>
</widget>

```

## 4.4. Аутентификация пользователей (Single-Sign-On)


Этот раздел описывает SSO функции, поддерживаемые Менеджером приложений, и отвечает на вопросы, как сохранять учетную информацию и как использовать ее в приложении.

### 4.4.1. Что такое система единого входа (Single-Sign-On)


Вводить регистрационные данные с помощью пульта управления достаточно неудобно, поэтому если бы пользователю приходилось каждый раз, когда он использует приложение, проделывать эту операцию, это было бы для него крайне утомительно. Для устранения этого неудобства Менеджер приложений использует систему единого входа (здесь и далее SSO). Менеджер приложений хранит учетную информацию, которую ввел пользователь, и передает ее в приложение. Это избавляет пользователя от необходимости постоянного ввода логина и пароля. Учетная информация кодируется и хранится в защищенном месте.

Используйте SSO, как описано ниже.


#### ① Создайте учетную запись пользователя (ТВ-аккаунт)

Для того чтобы использовать SSO, сначала необходимо создать учетную запись пользователя (ТВ-аккаунт). Такая запись в качестве пароля использует четырехзначный (платформа 2010) или шестизначный (2011) PIN-код. Учетная запись создается по кнопке  Вход -> Создание учетной записи.

#### ② Зарегистрируйте учетную запись службы провайдера

В созданной учетной записи пользователя (ТВ-аккаунте) зарегистрируйте учетную информацию службы провайдера (исключения: YouTube, Picasa). Учетная запись службы сохраняется только после того, как она была признана действующей. Для того чтобы зарегистрировать службу нажмите кнопку  Настройки -> Управление учетными записями.

#### ③ Войдите в систему (в ТВ-аккаунт)

Для входа в систему нажмите красную кнопку  на пульте управления (на экране внизу вы можете видеть подсказку) и введите логин и пароль, который вы создали в пункте ①. Регистрационные данные для большинства служб провайдеров могут быть сохранены в учетной записи пользователя (ТВ-аккаунте). Таким образом, войдя в свой ТВ-аккаунт, вы получаете доступ к этим данным, которые в некоторых приложениях могут быть использованы для аутентификации на сайте провайдера услуги.

#### ④ Запустите приложение


На этом этапе Менеджер приложений определяет, возможна ли для приложения аутентификация через SSO или нет. Если возможна, то Менеджер приложений передает регистрационные данные службы в приложение. Процесс регистрации учетной информации был описан в пункте ②.

#### ⑤ Использование учетной информации для аутентификации на сайте провайдера

Менеджер приложений передает регистрационные данные службы в приложение вместе с другими данными в виде строки, которая должна быть подвергнута грамматическому разбору.


Следующие разделы описывают, что должно быть реализовано в приложении, которое использует учетную информацию для доступа к службе провайдера.

#### 4.4.2. Регистрация учетной записи службы провайдера

Для того чтобы зарегистрировать учетную запись службы нажмите кнопку  Настройки -> Управление учетными записями.

Если вы хотите увидеть в списке имя службы, добавьте в файл config.xml строки по следующему образцу:

```
<srname>МояКомпания</srname>
<login>y</login>
```

После этого вы сможете найти имя МояКомпания в списке служб (в меню  Настройки -> Управление учетными записями). Если вы его не видите, повторно включите телевизор, возможно, Менеджер приложений требует перезагрузки.

Теперь может быть введена учетная запись для службы “МояКомпания”. Однако процесс регистрации учетной записи не может быть завершен, если отсутствует модуль, который отвечает за проверку и подтверждение достоверности введенной информации. Вам необходимо создать такой модуль.

Подтверждение достоверности необходимо, для того чтобы убедиться, что введенная учетная информация является полной (содержит логин и пароль) и позволяет получить доступ на действующий сайт провайдера. Если доступ к сайту через API (т.н. программный вход) может быть получен, тогда регистрационные данные признаются достоверными.

В Google за аутентификацию пользователей отвечает интерфейс ClientLogin. Если аутентификация прошла успешно, учетная запись признается корректной. Более подробную информацию о Client Login, вы можете получить по ссылке:

<http://code.google.com/intl/ko/apis/accounts/docs/AuthForInstalledApps.html>

Модуль проверки достоверности учетной информации создается по следующему сценарию.

- ① Укажите в файле config.xml имя js файла, в котором реализована функция проверки. Для этого используйте тэг <srauthjs>. Если файл называется Auth11101000000.js, то указывается только его имя Auth11101000000. Менеджер приложений будет выполнять функцию, содержащуюся в этом файле. Так как для всех приложений имена файлов должны быть уникальными, рекомендуется использовать в наименованиях идентификатор приложения (curWidget.id).

```
<srauthjs>Auth11101000000</srauthjs>
```

- ② Создайте файл с соответствующим именем, в котором будет содержаться функция проверки. Этот файл должен находиться в каталоге верхнего уровня, там же где и файл index.html.
- ③ Напишите функцию проверки достоверности регистрационных данных службы провайдера. Структура функции должна удовлетворять следующим требованиям:
  - Должен быть объявлен объект с именем, которое вы указали в пункте ①.
  - К объекту должна быть добавлена функция „checkAccount”.

- ④ Функция „checkAccount” должна осуществлять проверку достоверности учетной записи службы и передавать результат в функцию callback.

Пример

```
// Создаем объект
var Auth11101000000 = {
}
// Добавляем функцию checkAccount
Auth11101000000.checkAccount = function (id, pw, fnCallback) {
    /**
     * В качестве параметров из Менеджера приложений в функцию передаются ID и пароль
     * Проверяем id и пароль и вызываем функцию callback.
     */

    // ...
    // Получаем подтверждение корректности/некорректности учетной информации
    // ...

    // Возвращаем результат
    if (isValidAccount) {
        fnCallback ('TRUE');
    }
    else {
        fnCallback ('FALSE');
    }
}
}
```

Значения, передаваемые в функцию Callback, и их описание приведены ниже.

Возвращаемое значение	Описание
'TRUE'	ID и пароль достоверны. Менеджер приложений регистрирует их в ТВ-аккаунте.
'FALSE'	Регистрационные данные неверны. Менеджер приложений предложит пользователю ввести информацию заново.
'ERROR'	В процессе проверки произошла ошибка. Менеджер приложений признает, что произошла ошибка, и через какое-то время предложит пользователю попробовать сохранить регистрационные данные еще раз.
Другое	Значения, отличные от трех приведенных выше, аналогичны „ERROR”.

#### 4.4.3. Как в приложении получить регистрационные данные службы провайдера

В приложении можно получить доступ к учетной записи службы провайдера. Для этого необходимо выполнить следующие действия.

- ① Добавьте в файл config.xml строки, как показано ниже.

Это дает понять Менеджеру приложений, что в приложении требуется аутентификация для доступа к службе провайдера, которая называется “МояКомпания”.

```
<cpname>МояКомпания</cpname>  
<login>y</login>
```

## ② Войдите в ТВ-аккаунт

Только после этого учетная информация, зарегистрированная в вашем ТВ-аккаунте, станет доступной.

## ③ Запустите приложение

Регистрационные данные службы МояКомпания должны храниться в вашей учетной записи (ТВ-аккаунте). Как правило, в учетной записи пользователя (ТВ-аккаунте) регистрируется несколько служб провайдеров, и приложение получает доступ к той службе, которая указана в файле config.xml (см. пункт ①).

## ④ Как прочитать ID и пароль

Если соблюдены все вышеперечисленные условия, ID и пароль передаются в глобальную переменную браузера window.location.search. Эта переменная содержит различную информацию, которая может быть получена путем синтаксического разбора. Подробнее о window.location.search смотрите в разделе 6.7.

## 4.5. Переустановка (Reset)

Функция переустановки отвечает за инициализацию/удаление приложения.

### 4.5.1. Что такое переустановка

Это функция, которая в момент удаления или инициализации приложения в Менеджере приложений запускает модуль инициализации, реализованный в этом приложении.

### 4.5.2. Как это работает

1. Менеджер приложений в момент удаления/инициализации приложения читает информацию из файла config.xml установленного приложения.
2. Проверяет, содержит ли файл config.xml приложения информацию о модуле, отвечающем за инициализацию.
3. Если такой модуль указан, получает информацию об этом модуле.
4. Выполняет модуль в соответствии с установленными правилами и пытается инициализировать приложение.

\* Менеджер приложений выполняет модуль инициализации, не получая от него возвращаемого значения.

### 4.5.3. Что для этого должны реализовать разработчики

1. Добавить необходимую информацию в файл config.xml (см. раздел 4.5.4).
2. Реализовать модуль инициализации (см. раздел 4.5.5).

#### 4.5.4. Инструкция по реализации config.xml

Добавьте в файл элемент как показано ниже.

```
<deleteJS>MyReset</deleteJS>
```

\*Без этого тэга функция инициализации приложения выполнена не будет.

#### 4.5.5. Инструкция по реализации модуля инициализации

1. Имя файла должно совпадать с именем класса.
2. Без параметров.
3. Имя функции должно быть "reset".

```
var MyReset= {
...
}
MyReset.reset = function() {
...
...
alert("Reset Complete!");
}
```

#### 4.6.Пример взаимодействия с объектом XMLHttpRequest

В этом разделе мы рассмотрим простой пример взаимодействия с объектом XHR (XMLHttpRequest), который является ключевым элементом AJAX технологии.

Ниже приведен пример кода HTML, который содержит файл XHRExample.js и по событию „load“ выполняет функцию XHRExample.onload().

```
<!DOCTYPE html>

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>XHR Example</title>
  <script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/TVKeyValue.js"></script>
  <script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/Widget.js"></script>
  <script type="text/javascript" language="javascript" src="XHRExample.js"></script>
</head>

<body onload="XHRExample.onload()">
  <div id='intro'>XHR Example.</div>
</body>
</html>
```

Ниже приведен пример JavaScript кода. XHRExample объявлен как объект и содержит переменную XHRObj. Если XHRObj уже существует, XHRExample вызывает функцию destroy(), которую далее мы рассмотрим более подробно. Если объект XHR успешно создан, присваиваем его переменной XHRObj и запрашиваем данные.

Если состояние объекта XHR изменяется на 4, что означает, что процесс получения данных завершен, выполняется функция recieveData(), которая указана в свойстве onreadystatechange.

```
var XHRExample = {
    XHRObj : null
}
var tvKey = new Common.API.TVKeyValue();
var widgetAPI = new Common.API.Widget();

XHRExample.onload = function () {
    widgetAPI.sendReadyEvent();
    var URL = ""; // Test URL here

    if (this.XHRObj != null)
        this.XHRObj.destroy();
    this.XHRObj = new XMLHttpRequest();

    if (this.XHRObj) {
        this.XHRObj.onreadystatechange = function () {
            if (XHRExample.XHRObj.readyState == 4) {
                XHRExample.recieveData();
            }
        };
        this.XHRObj.open("GET", URL, true);
        this.XHRObj.send(null);
    }
}
XHRExample.recieveData = function () {
    alert(this.XHRObj.responseText);
}
```

В приведенном выше примере следует обратить внимание на два момента. Первый - это то, что переменная XHRObj находится внутри XHRExample объекта. Второй - то, что функция destroy() вызывается перед тем, как переменной XHRObj присваивается объект XMLHttpRequest.

Если вы загружаете оперативную память, постоянно создавая и используя объект XHR, то приложения, установленные в вашем телевизоре, в какой-то момент времени могут перестать работать. Чтобы избежать этой ситуации, перед тем как создать новый объект XHR, вы прежде должны удалить объект, который вы использовали ранее. Чтобы освободить ранее использовавшийся объект, необходимо хранить ссылку на него. Поэтому разработчики должны использовать переменные для работы с XHR объектами.

Если вы хотите удалить XHR объект из памяти, используйте функцию destroy() этого объекта, которая поддерживается браузером App Engine для того, чтобы вы могли более рационально использовать память телевизора.

## XHRObject.destroy

Удаляет созданный XHR объект из памяти.

<b>Синтаксис</b>	XHRObject.destroy()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Эта функция, предназначенная для оптимизации использования памяти, доступна только в браузере App Engine для пользователей Samsung TV.
<b>Пример</b>	<pre>if (this.XHRObj != null)     this.XHRObj.destroy(); this.XHRObj = new XMLHttpRequest();</pre>

В приведенном выше примере кода используются два общих модуля-библиотеки Менеджера приложений. Для получения более детальной информации, обратитесь к разделу 5 Общие модули.

## 5. Общие модули

В этом разделе содержится обзор библиотек, которые используются в приложениях Samsung Smart TV.

### 5.1. Общие модули Менеджера приложений

Менеджер приложений содержит общие модули (библиотеки), которые вы можете использовать в приложениях. Эти модули реализуют базовую функциональность и обеспечивают нормальное функционирование приложения на экране телевизора. Кроме этого, используя эти модули, приложения могут распознавать и обрабатывать нажатие кнопок пульта управления, использовать плагины и взаимодействовать с Менеджером приложений наиболее удобным и простым способом.

### 5.2. Как использовать общие модули

- 1 Включите общий модуль в файл index.html. Чтобы исключить возможность ошибок, которые может повлечь за собой нарушение порядка вызова функций, рекомендуется включать общие модули в начале документа. Для указания пути к общим модулям Менеджера приложений используется переменная окружения '\$MANAGER\_WIDGET'.

\*В SDK '\$MANAGER\_WIDGET' указывает на папку "commonlib", в которую устанавливается SDK.

```
<script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/TVKeyValue.js"></script>
```

- 2 Создайте объект общего модуля.

```
var tvKey = new Common.API.TVKeyValue();
```

- 3 Используйте атрибуты и методы этого объекта.

```
alert(tvKey.KEY_LEFT);
```

### 5.3. TVKeyValue

TVKeyValue	
Содержит значения кодов кнопок пульта управления.	
Путь к библиотеке	\$MANAGER_WIDGET/Common/API/TVKeyValue.js
Синтаксис	new Common.API.TVKeyValue()
Параметры	None
Возвращаемое значение	TVKeyValue object

Примечание	<p>Этот объект в качестве атрибутов содержит коды кнопок пульта управления телевизора. Используя эти атрибуты в приложении можно идентифицировать кнопки, которые нажимает пользователь.</p> <p>(Начиная с Since Samsung Smart TV version 2.265)</p>
Пример	<pre>var tvKey = new Common.API.TVKeyValue(); switch (KEY_CODE) {     case tvKey.KEY_LEFT:         // Do something         break;     case tvKey.KEY_RIGHT:         // Do something         break;     default:         break; }</pre>
Атрибуты	<ul style="list-style-type: none"> <li>▣ KEY_TOOLS</li> <li>▣ KEY_MUTE</li> <li>▣ KEY_RETURN</li> <li>▣ KEY_UP</li> <li>▣ KEY_DOWN</li> <li>▣ KEY_LEFT</li> <li>▣ KEY_RIGHT</li> <li>▣ KEY_WHEELDOWN</li> <li>▣ KEY_WHEELUP</li> <li>▣ KEY_ENTER</li> <li>▣ KEY_INFO</li> <li>▣ KEY_EXIT</li> <li>▣ KEY_RED</li> <li>▣ KEY_GREEN</li> <li>▣ KEY_YELLOW</li> <li>▣ KEY_BLUE</li> <li>▣ KEY_INFOLINK</li> <li>▣ KEY_RW</li> <li>▣ KEY_PAUSE</li> <li>▣ KEY_FF</li> <li>▣ KEY_PLAY</li> <li>▣ KEY_STOP</li> <li>▣ KEY_1</li> <li>▣ KEY_2</li> <li>▣ KEY_3</li> <li>▣ KEY_4</li> <li>▣ KEY_5</li> <li>▣ KEY_6</li> <li>▣ KEY_7</li> <li>▣ KEY_8</li> <li>▣ KEY_9</li> <li>▣ KEY_0</li> <li>▣ KEY_EMPTY</li> <li>▣ KEY_PRECH</li> <li>▣ KEY_SOURCE</li> <li>▣ KEY_CHLIST</li> <li>▣ KEY_MENU</li> <li>▣ KEY_WLINK</li> <li>▣ KEY_CC</li> </ul>

	<ul style="list-style-type: none"> <li>▫ KEY_CONTENT</li> <li>▫ KEY_FAVCH</li> <li>▫ KEY_REC</li> <li>▫ KEY_EMODE</li> <li>▫ KEY_DMA</li> <li>▫ KEY_PANEL_CH_UP</li> <li>▫ KEY_PANEL_CH_DOWN</li> <li>▫ KEY_PANEL_VOL_UP</li> <li>▫ KEY_PANEL_VOL_DOWN</li> <li>▫ KEY_PANEL_ENTER</li> <li>▫ KEY_PANEL_SOURCE</li> <li>▫ KEY_PANEL_MENU</li> <li>▫ KEY_PANEL_POWER</li> </ul>
<b>Методы</b>	None

## 5.4. Widget

Этот объект содержит функции, обеспечивающие нормальное функционирование приложения, такие как функция, которая оповещает Менеджер приложений о готовности приложения к запуску, функция регистрации/отключения кнопок, которые могут использовать пользователи для управления приложением, и другие.

<b>Widget</b>	
Содержит функции, необходимые для корректной работы приложения	
<b>Путь к библиотеке</b>	\$MANAGER_WIDGET/Common/API/Widget.js
<b>Синтаксис</b>	<code>new Common.API.Widget()</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	Widget object
<b>Примечание</b>	(Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var widgetAPI = new Common.API.Widget(); widgetAPI. sendReadyEvent();</pre>
<b>Атрибуты</b>	None
<b>Методы</b>	<ul style="list-style-type: none"> <li>▫ sendReadyEvent</li> <li>▫ sendExitEvent</li> <li>▫ sendReturnEvent</li> <li>▫ blockNavigation</li> <li>▫ putInnerHTML</li> <li>▫ getChannelWidgetListPath</li> <li>▫ getSearchWidgetListPath</li> <li>▫ runSearchWidget</li> </ul>

### 5.4.1. sendReadyEvent()

<b>sendReadyEvent</b>	
Оповещает Менеджер приложений о готовности приложения к запуску. Это необходимо для того, чтобы приложение могло быть отображено и запущено на экране телевизора.	
<b>Синтаксис</b>	sendReadyEvent()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	(Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var widgetAPI = new Common.API.Widget(); widgetAPI.sendReadyEvent();</pre>
<b>Enumeration</b>	None

### 5.4.2. sendExitEvent()

<b>sendExitEvent</b>	
Выполняет действие аналогичное нажатию кнопки Exit. Завершает работу приложения и возвращает пользователя на экран телевизора.	
<b>Синтаксис</b>	sendExitEvent()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	(Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var widgetAPI = new Common.API.Widget(); widgetAPI.sendExitEvent ();</pre>
<b>Enumeration</b>	None

### 5.4.3. sendReturnEvent()

<b>sendReturnEvent</b>	
Выполняет действие аналогичное нажатию кнопки Return. Завершает работу приложения и возвращает пользователя на экран Менеджера Приложений.	
<b>Синтаксис</b>	sendReturnEvent()
<b>Параметры</b>	None

<b>Возвращаемое значение</b>	None
<b>Примечание</b>	(Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<code>var widgetAPI = new Common.API.Widget(); widgetAPI.sendReturnEvent ();</code>
<b>Enumeration</b>	None

#### 5.4.4. blockNavigation()

<b>blockNavigation</b>	
Отключает несанкционированный выход из приложения.	
<b>Синтаксис</b>	blockNavigation(keyEvent)
<b>Параметры</b>	keyEvent: Key event
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	События KEY_RETURN и KEY_EXIT прерывают выполнение приложения. Эта функция предотвращает несанкционированный выход из приложения, выполнение приложения продолжается. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>function keyDown () {     switch(event.keyCode) {         case this.tvKey.KEY_LEFT:             // Do something             break;         case this.tvKey.KEY_RIGHT:             // Do something             break;         case this.tvKey.KEY_RETURN:             widgetAPI.blockNavigation(event);             // Not terminated             break;         case this.tvKey.KEY_EXIT:             // Terminated by force             break;         default:             break;     } }</pre>
<b>Enumeration</b>	None

## 5.4.5. putInnerHTML()

putInnerHTML	
Функция innerHTML, позволяющая оптимально использовать оперативную память телевизора.	
Синтаксис	putInnerHTML(div, contents)
Параметры	div: Div Element to use innerHTML contents: Things to insert in innerHTML
Возвращаемое значение	None
Примечание	Рекомендуется использовать эту функцию, поскольку многократное использование свойства innerHTML может привести к переполнению памяти телевизора. (Начиная с Samsung Smart TV version 2.265)
Пример	<pre>var widgetAPI = new Common.API.Widget(); var divElement = document.getElementById("divID"); var contents = "Text to change";  widgetAPI.putInnerHTML(divElement, contents);</pre>
Enumeration	None

## 5.4.6. getChannelWidgetListPath()

getChannelWidgetListPath	
Method that brings the path of xml file having the child application list of the current channel bound application	
Синтаксис	getChannelWidgetListPath()
Параметры	None
Возвращаемое значение	string : xml file path
Примечание	This method is used when the channel bound application needs information on the currently installed child application. (Начиная с Samsung Smart TV version 2.265)
Пример	<pre>var widgetAPI = new Common.API.Widget(); var xmlPath = widgetAPI.getChannelWidgetListPath();</pre>
Enumeration	None

5.4.7. `getSearchWidgetListPath()`

<b>getSearchWidgetListPath</b>	
Возвращает путь к xml файлу, в котором содержится список связанных приложений	
<b>Синтаксис</b>	<code>getSearchWidgetListPath()</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	string : xml file path
<b>Примечание</b>	Эта функция используется в приложениях, для которых в файле config.xml в тэге search установлено значение "y". (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var widgetAPI = new Common.API.Widget(); var xmlPath = widgetAPI.getSearchWidgetListPath();</pre>
<b>Enumeration</b>	None

5.4.8. `runSearchWidget()`

<b>runSearchWidget</b>	
Вызывает в приложении другое приложение	
<b>Синтаксис</b>	<code>runSearchWidget (appId, extraInfo)</code>
<b>Параметры</b>	appId : ID приложения, которое необходимо выполнить extraInfo : Строка, которая передается в вызываемое приложение
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	This method is used when the channel bound application requires information on the installed child application at present. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var widgetAPI = new Common.API.Widget(); widgetAPI.runSearchWidget("111011000001", "key1=value1&amp;key2=value2");</pre>
<b>Enumeration</b>	None

5.4.9. `checkSapTicket ()`

<b>checkSapTicket</b>	
Method called when checking to have the valid ticket for current logged-in user.	
<b>Синтаксис</b>	<code>checkSapTicket ()</code>

Параметры	None
Возвращаемое значение	None
Примечание	You should register event handler for passing the return value. If you have valid ticket, the return value is “stat=ok&ticket=686a8281-e952-4dcc-a67b-57e8f92e5530” Otherwise, it is “stat=fail&ticket=null”. (Начиная с Samsung Smart TV version 3.037)
Пример	<pre>// Register Event Handler curWidget.onWidgetEvent = AAAA; var widgetAPI = new Common.API.Widget(); widgetAPI.checkSapTicket(); function AAAA(event){ if( event.type == Common.API.EVENT_ENUM.PNS_CHECK_TICKET){ // The return value is event.data. // TODO } }</pre>
Enumeration	None

#### 5.4.10. requestSapTicket ()

requestSapTicket	
Method called when requesting new Sap Ticket for Application Server.	
Синтаксис	requestSapTicket ()
Параметры	None
Возвращаемое значение	None
Примечание	You should register event handler for passing the return value. If new ticket is generated normally, the return value is “stat=ok&ticket=686a8281-e952-4dcc-a67b-57e8f92e5530” Otherwise, it is “stat=fail&ticket=null”. (Начиная с Samsung Smart TV version 3.037)
Пример	<pre>// Register Event Handler curWidget.onWidgetEvent = AAAA; var widgetAPI = new Common.API.Widget(); widgetAPI.requestSapTicket(); function AAAA(event){ if( event.type == Common.API.EVENT_ENUM.PNS_REQUEST_TICKET){ // The return value is event.data. // TODO } }</pre>
Enumeration	None

## 5.5. Plugin

Менеджер приложений содержит класс-обертку, который делает использование некоторых встраиваемых функций (плагинов) более удобным. Для этого вам надо просто включить в файл index.html соответствующие объекты данного класса, для каждого из которых Менеджер приложений поддерживает свой набор методов.

Plugin	
Объект для работы со встроенными функциями (плагинами).	
Путь к библиотеке	\$MANAGER_WIDGET/Common/API/Plugin.js
Синтаксис	<code>new Common.API.Plugin()</code>
Параметры	None
Возвращаемое значение	Plugin object
Примечание	(Начиная с Samsung Smart TV version 2.265)
Пример	<code>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOnWatchDog();</code>
Атрибуты	None
Методы	<p>Object ID : <a href="#">pluginObjectTVMW</a></p> <ul style="list-style-type: none"> <li>▫ setOnWatchDog</li> <li>▫ setOffWatchDog</li> <li>▫ registKey</li> <li>▫ unregistKey</li> <li>▫ registIMEKey</li> <li>▫ unregistIMEKey</li> <li>▫ registAllKey</li> <li>▫ unregistAllKey</li> <li>▫ registFullWidgetKey</li> <li>▫ registPartWidgetKey</li> </ul> <p>Object ID : <a href="#">pluginObjectVideo</a></p> <ul style="list-style-type: none"> <li>▫ setOnOSDState</li> <li>▫ setOffOSDState</li> </ul> <p>Object ID : <a href="#">pluginObjectNNAvi</a></p> <ul style="list-style-type: none"> <li>▫ SetBannerState</li> <li>▫ ShowTools</li> <li>▫ setOnIdleEvent</li> <li>▫ setOffIdleEvent</li> <li>▫ setOnScreenSaver</li> <li>▫ setOffScreenSaver</li> </ul> <p>Object ID : <a href="#">pluginObjectAppCommon</a></p> <ul style="list-style-type: none"> <li>▫ isViewerKey</li> </ul>

Список объектов	<div> <div> <div>□ pluginObjectTVMW</div> <div>&lt;OBJECT id='pluginObjectTVMW' border=0 classid='clsid:SAMSUNG-INFOLINK-TVMW' style='opacity:0.0;background-color:#000000;width:0px;height:0px;'&gt;&lt;/OBJECT&gt;</div> </div> <div> <div>□ pluginObjectVideo</div> <div>&lt;OBJECT id='pluginObjectVideo' border=0 classid='clsid:SAMSUNG-INFOLINK-VIDEO' style='opacity:0.0;background-color:#000000;'&gt;&lt;/OBJECT&gt;</div> </div> <div> <div>□ pluginObjectNNAvi</div> <div>&lt;OBJECT id='pluginObjectNNAvi' classid='clsid:SAMSUNG-INFOLINK-NNAVI' style='opacity:0.0;background-color:#000000;width:0px;height:0px;'&gt;&lt;/OBJECT&gt;</div> </div> <div> <div>□ pluginObjectAppCommon</div> <div>&lt;OBJECT id='pluginObjectAppCommon' classid='clsid:SAMSUNG-INFOLINK-APPCOMMON' style='opacity:0.0;background-color:#000000;width:0px;height:0px;'&gt;&lt;/OBJECT&gt;</div> </div> </div>
-----------------	--

### 5.5.1. setOnWatchDog()

setOnWatchDog	
Включает контрольный таймер Watch dog	
Синтаксис	setOnWatchDog()
Параметры	None
Возвращаемое значение	None
Примечание	(Since Samsung Smart TV version 2.265)
Пример	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOnWatchDog();</pre>
Enumeration	None

### 5.5.2. setOffWatchDog()

setOffWatchDog	
Выключает контрольный таймер Watch dog	
Синтаксис	setOffWatchDog()
Параметры	None
Возвращаемое значение	None
Примечание	(Начиная с Samsung Smart TV version 2.265)
Пример	<pre>var pluginAPI = new Common.API.Plugin();</pre>

	<code>pluginAPI.setOffWatchDog();</code>
Enumeration	None

### 5.5.3. setOnOSDState()

setOnOSDState	
Устанавливает защищенную область OSD.	
Синтаксис	<code>setOnOSDState(left, top, width, height)</code>
Параметры	left : левая координата области top : верхняя координата области width : ширина области height : высота области
Возвращаемое значение	None
Примечание	(Начиная с Samsung Smart TV version 2.265)
Пример	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOnOSDState(0,0, 330, 540);</pre>
Enumeration	None

### 5.5.4. setOffOSDState()

setOffOSDState	
Отменяет установленную ранее область OSD.	
Синтаксис	<code>setOffOSDState(left, top, width, height)</code>
Параметры	left : левая координата области top : верхняя координата области width : ширина области height : высота области
Возвращаемое значение	None
Примечание	(Начиная с Samsung Smart TV version 2.265)
Пример	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOffOSDState(0,0, 330, 540);</pre>
Enumeration	None

### 5.5.5. registKey()

registKey
-----------

Позволяет Менеджеру приложений зарегистрировать определенную кнопку пульта управления	
<b>Синтаксис</b>	<code>registKey(pNumKeyCode)</code>
<b>Параметры</b>	<code>pNumKeyCode</code> : код кнопки
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Функция регистрации/отключения кнопок должна быть вызвана после события <code>window.onshow</code> . См. раздел 6.4 Кнопки пульта управления. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.registKey(tvKey.KEY_ TOOLS);</pre>
<b>Enumeration</b>	None

#### 5.5.6. `unregistKey()`

<b>unregistKey</b>	
Позволяет Менеджеру приложений зарегистрировать определенную кнопку пульта управления	
<b>Синтаксис</b>	<code>unregistKey(pNumKeyCode)</code>
<b>Параметры</b>	<code>pNumKeyCode</code> : код кнопки
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Функция регистрации/отключения кнопок должна быть вызвана после события <code>window.onshow</code> . См. раздел 6.4 Кнопки пульта управления. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.unregistKey(tvKey.KEY_ TOOLS);</pre>
<b>Enumeration</b>	None

#### 5.5.7. `registIMEKey()`

<b>registIMEKey</b>	
Позволяет Менеджеру приложений зарегистрировать IME кнопку	
<b>Синтаксис</b>	<code>registIMEKey()</code>
<b>Параметры</b>	<code>pNumKeyCode</code> : код кнопки
<b>Возвращаемое значение</b>	None

<b>Примечание</b>	Регистрирует кнопки от 0 до 9, „-„(дефис), и кнопку previous channel, которые используются для ввода в редакторе IME. Функция регистрации/отключения кнопок должна быть вызвана после события window.onshow. См. раздел 6.4 Кнопки пульта управления. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.unregistIMEKey();</pre>
<b>Enumeration</b>	None

#### 5.5.9. registAllKey

<b>registAllKey</b>	
Позволяет Менеджеру приложений зарегистрировать все кнопки пульта управления	
<b>Синтаксис</b>	registAllKey()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Регистрирует все кнопки Функция регистрации/отключения кнопок должна быть вызвана после события window.onshow. См. раздел 6.4 Кнопки пульта управления. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.registAllKey();</pre>
<b>Enumeration</b>	None

#### 5.5.10. unregistAllKey()

<b>unregistAllKey</b>	
Позволяет Менеджеру приложений отключить все кнопки пульта управления	
<b>Синтаксис</b>	unregistAllKey()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Отключает все кнопки Функция регистрации/отключения кнопок должна быть вызвана после события window.onshow. См. раздел 6.4 Кнопки пульта управления.

	(Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<code>var pluginAPI = new Common.API.Plugin(); pluginAPI.unregisterAllKey();</code>
<b>Enumeration</b>	None

#### 5.5.11. `registFullWidgetKey()`

<b>registFullWidgetKey</b>	
Позволяет Менеджеру приложений зарегистрировать набор определенных кнопок пульта управления	
<b>Синтаксис</b>	<code>registFullWidgetKey()</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Регистрирует набор кнопок для полноэкранного приложения Функция регистрации/отключения кнопок должна быть вызвана после события <code>window.onshow</code> . См. раздел 6.4 Кнопки пульта управления. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<code>var pluginAPI = new Common.API.Plugin(); pluginAPI.registFullWidgetKey();</code>
<b>Enumeration</b>	None

#### 5.5.12. `registPartWidgetKey()`

<b>registPartWidgetKey</b>	
Позволяет Менеджеру приложений зарегистрировать набор определенных кнопок пульта управления	
<b>Синтаксис</b>	<code>registPartWidgetKey ()</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Регистрирует набор кнопок для приложения, занимающего часть экрана. Функция регистрации/отключения кнопок должна быть вызвана после события <code>window.onshow</code> . См. раздел 6.4 Кнопки пульта

	управления. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<code>var pluginAPI = new Common.API.Plugin(); pluginAPI.registPartWidgetKey ();</code>
<b>Enumeration</b>	None

### 5.5.13. SetBannerState()

<b>SetBannerState</b>	
Используется, если вы хотите показывать в приложении стандартные экранные меню (OSD) управления каналами и громкостью.	
<b>Синтаксис</b>	SetBannerState (nState)
<b>Параметры</b>	nState : установленное значение для экранного меню (OSD), которое вы хотите использовать в приложении. PL_NNAVI_STATE_BANNER_NONE = 0; PL_NNAVI_STATE_BANNER_VOL = 1; PL_NNAVI_STATE_BANNER_VOL_CH = 2;
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Если вы хотите вывести на экран стандартное экранное меню (OSD) настроек громкости, кнопки управления громкостью должны быть зарегистрированы. Аналогично для каналов. См. пример. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<code>var PL_NNAVI_STATE_BANNER_NONE = 0; var PL_NNAVI_STATE_BANNER_VOL = 1; var PL_NNAVI_STATE_BANNER_VOL_CH = 2; var pluginAPI = new Common.API.Plugin(); var tvKey = new Common.API.TVKeyValue(); pluginAPI.unregistKey(tvKey.KEY_VOL_UP); pluginAPI.unregistKey(tvKey.KEY_VOL_DOWN); pluginAPI.SetBannerState(PL_NNAVI_STATE_BANNER_VOL);</code>
<b>Enumeration</b>	None

### 5.5.14. ShowTools()

<b>ShowTools</b>	
Помещает на экран всплывающие окна настройки звука и изображения.	
<b>Синтаксис</b>	ShowTools (nTool)
<b>Параметры</b>	nTool: установленное значение код для всплывающего окна, которое вы хотите поместить на экран.

	0 : Sound Setting popup 1 : Picture Setting popup
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Вы можете отслеживать момент, когда всплывающее окно будет закрыто, чтобы обработать это событие в функции curWidget.onWidgetEvent. (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>curWidget.onWidgetEvent = function(){ // код, который будет выполнен после закрытия всплывающего окна } var pluginAPI = new Common.API.Plugin(); pluginAPI.ShowTools(1);</pre>
<b>Enumeration</b>	None

#### 5.5.15. setOnIdleEvent()

<b>setOnIdleEvent</b>	
Эта функция устанавливает значение „Idle ON” с целью закрытия приложения в случае отсутствия в течение длительного времени реакции со стороны пользователя.	
<b>Синтаксис</b>	setOnIdleEvent()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	1. По умолчанию установлено значение „Idle ON”. 2. Вызывается для восстановления процесса отслеживания бездействия системы, если ранее была вызвана функция setOffIdleEvent (искл.: Когда останавливается воспроизведение изображений). (Начиная с Samsung Smart TV version 2.265)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOnIdleEvent();</pre>
<b>Enumeration</b>	None

#### 5.5.16. setOffIdleEvent()

<b>setOffIdleEvent</b>	
Эта функция устанавливает значение „Idle OFF” с целью предотвратить закрытие приложения в случае отсутствия в течение длительного времени реакции со стороны пользователя.	
<b>Синтаксис</b>	setOffIdleEvent()

Параметры	None
Возвращаемое значение	None
Примечание	1. По умолчанию установлено значение „Idle ON”. 2. Установка „Idle OFF” необходима, если в приложении предусмотрен просмотр видео/изображений в течение длительного времени. (Начиная с Samsung Smart TV version 2.265)
Пример	<code>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOffIdleEvent();</code>
Enumeration	None

#### 5.5.17. setOnScreenSaver()

<b>setOnScreenSaver</b>	
Активирует экранную заставку.	
Синтаксис	<code>setOnScreenSaver()</code>
Параметры	None
Возвращаемое значение	None
Примечание	Если приложение закрыто, экранная заставка неактивна. (Начиная с Samsung Smart TV version 2.265)
Пример	<code>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOnScreenSaver();</code>
Enumeration	None

#### 5.5.18. setOffScreenSaver()

<b>setOffScreenSaver</b>	
Убирает экранную заставку.	
Синтаксис	<code>setOffScreenSaver()</code>
Параметры	None
Возвращаемое значение	None
Примечание	Вызывается после того, как была вызвана функция <code>etOnScreenSaver</code> . (Начиная с Samsung Smart TV version 2.265)
Пример	<code>var pluginAPI = new Common.API.Plugin();</code>

	<code>pluginAPI.setOffScreenSaver();</code>
<b>Enumeration</b>	None

#### 5.5.19. `isViewerKey()`

<b>isViewerKey</b>	
Функция для определения, относится ли нажатая кнопка к приложению или просмотру ТВ.	
<b>Синтаксис</b>	<code>isViewerKey (pKeyCode)</code>
<b>Параметры</b>	<code>pKeyCode</code> : текущая нажатая кнопка пульта управления
<b>Возвращаемое значение</b>	True - Кнопка ТВ просмотра False - Кнопка для управления приложением
<b>Примечание</b>	When the TV viewer needs to proceed the entered remote control keys, the inputted value in TV viewer is automatically transferred (Начиная с Samsung Smart TV version 2.266)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.isViewerKey(KEY_ENTER)</pre>
<b>Enumeration</b>	None

#### 5.5.20. `setOnFullScreen ()`

<b>setOnFullScreen</b>	
Устанавливает полноэкранный режим.	
<b>Синтаксис</b>	<code>setOnFullScreen()</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Если в файле config.xml установлено значение <code>&lt;fullwidget&gt;y&lt;/fullwidget&gt;</code> , использовать эту функцию нет необходимости. (Начиная с Samsung Smart TV version 2.306)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOnFullScreen();</pre>
<b>Enumeration</b>	None

#### 5.5.21. `setOffFullScreen()`

<b>setOffFullScreen</b>
-------------------------

Отменяет полноэкранный режим.	
<b>Синтаксис</b>	setOffFullScreen()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Вызывается после того, как была вызвана функция setOnFullScreen. (Начиная с Samsung Smart TV version 2.306)
<b>Пример</b>	<pre>var pluginAPI = new Common.API.Plugin(); pluginAPI.setOffFullScreen();</pre>
<b>Enumeration</b>	None

## 5.6. CImageViewer

Этот раздел содержит описание общего модуля (библиотеки) „CImageViewer.

### 5.6.1. Что такое CImageViewer

CImageViewer - это общий модуль, который делает доступным воспроизведение файлов формата JPEG на экране цифровых телевизоров SAMSUNG. Благодаря этому модулю на экране телевизора изображения могут отображаться с высоким разрешением 1920 x 1080 пикселей. Кроме этого изображения масштабируются в соответствии с размерами области, которую вы для них определите, а также к ним могут применяться различные анимационные эффекты.

Если вы хотите использовать CImageViewer, то в качестве источника сигнала должен быть установлен Медиа-источник (Media source). Переключение источника может быть выполнено путем установки значения “у” в тэге <srcctl> в файле config.xml или путем вызова функции SetMediaSource() плагина TVMW Plugin.

### 5.6.2. Спецификации

Характеристика	Описание
<b>Разрешение</b>	1920 x 1080 пикселей Изображения отображаются в Медиа-источнике. Для приложения определено разрешение 960 x 540 пикселей, но для изображений, отображаемых в медиа-источнике, устанавливается разрешение 1920 x 1080 пикселей.
<b>Поддерживаемые форматы</b>	только JPEG
<b>Диапазон масштабирования</b>	Изображение масштабируется в соответствии с размерами определенной вами области, внутри которой оно будет воспроизводиться. В этом случае, разрешение должно превышать 300 пикселей. Если установить разрешение меньше 300 пикселей, то изображение может быть искажено или обрезано.

<b>Анимационные эффекты межкадрового преобразования</b>	Всего поддерживаются 10 режимов. (NONE, FADE1, FADE2, BLIND, SPIRAL, CHECKER, LINEAR, STAIRS, WIPE, RANDOM) Для получения более детальной информации см. раздел 5.6.4.
---	--

### 5.6.3. Базовые инструкции

Касаются расположения и показа изображений с использованием масштабирования.

- ① Подключите общий модуль путем добавления в файле index.html к тэгу <head> следующего кода

```
<script type="text/javascript" language="javascript"
src="$MANAGER_WIDGET/Common/API/CImageViewer.js"></script>
```

- ② Создайте экземпляр объекта ImageViewer.

```
var ImageViewer = new CImageViewer('Common ImageViewer');
```

- ③ Инициализация ImageViewer.

Определите область, внутри которой будут воспроизводиться изображения, а также функции для обработки каждого события. Функция show() служит для отображения области, которой на экране будет управлять объект ImageViewer.

Эта область устанавливается на базе стандартного разрешения экрана (960x540 pixel) путем вызова функции setFrameArea(). Внутри выделенной области действует разрешение 1920x1080 пикселей.

- ④ Воспроизведение изображений.

Изображения сохраняют свои оригинальные пропорции и масштабируются в соответствии с указанными размерами.

```
ImageViewer.play('$WIDGET/picture.jpg', 640, 480);
```

- ⑤ Завершение приложения

Когда приложение завершается, объект ImageViewer должен быть удален из памяти путем вызова функции destroy(). В противном случае может возникнуть критическая проблема, такая как приостановка работы ТВ системы, и могут возникнуть проблемы в других приложениях, которые используют ImageViewer.

```
// Вызов ImageViewer.destroy() в функции <body>'s onunload
ImageViewer.destroy();
```

### 5.6.4. Анимационные эффекты при межкадровых переходах (Transition Effect)

При смене изображений (кадров) могут воспроизводиться специальные анимационные эффекты. Эти эффекты затрагивают не только область изображения, но и весь экран. Следующая таблица иллюстрирует 10 доступных для использования эффектов преобразования кадра 1 в кадр 2.

Effect	Description
NONE	Эффект отсутствует.
FADE1	Кадр 1 постепенно выцветает и начинает проявляться кадр 2.
FADE2	Кадр 1 постепенно выцветает до белого и начинает проявляться кадр 2.
BLIND	Эффект жалюзи.
SPIRAL	Кадр 2 появляется, раскручиваясь по часовой стрелке.
CHECKER	Кадр 2 постепенно появляется в виде решетки.
LINEAR	Каждая линия Кадра 1 заменяется линией Кадра 2 слева направо.
STAIRS	Кадр 1 постепенно трансформируется в кадр 2 в форме ступенек от верхнего левого угла к нижнему правому.
WIPE	Переход происходит, когда центр кадра растягивается вверх и вниз, или верхняя и нижняя стороны вытягиваются к центру.
RANDOM	Эффекты, перечисленные выше (кроме NONE), накладываются в случайном порядке.

Можно использовать свойства атрибута Effect как объекта ImageViewer, так и класса CImageViewer.

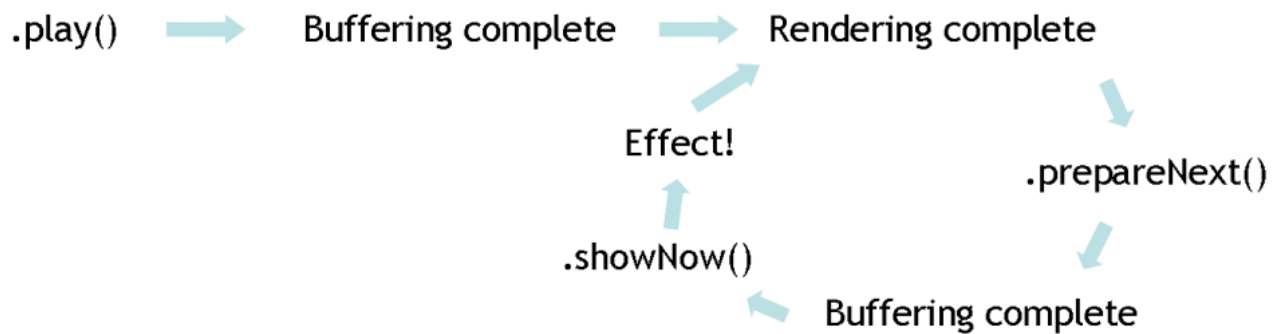
```
var ImageViewer = new CImageViewer('Common ImageViewer');
ImageViewer.Effect.FADE2
or
CImageViewer.Effect.FADE2
```

#### 5.6.4.1. Основные принципы создания слайд-шоу

Слайд-шоу выполняется в соответствии со следующим сценарием:

Показ первого изображения □ Подготовка второго изображения □ Показ второго изображения □  
Подготовка третьего изображения .....

Для воспроизведения первого изображения вызывается функция .play(). Далее происходит событие *Завершение буферизации*, затем событие *Завершение визуализации*, которое сигнализирует о том, что изображение полностью отобразилось на экране. Затем вызывается функция prepareNext() для того, чтобы подготовить к показу следующее изображение. В этот момент никаких изменений не происходит, т.к. эта подготовка выполняется в фоновом режиме. О готовности изображения сообщает событие *Завершение буферизации*. Начиная с этого момента, может быть применен анимационный эффект путем вызова функции .showNow(). После того, как анимационный эффект завершен, происходит событие *Завершение визуализации* и переход к следующему изображению.



#### 5.6.4.2. Как реализовать анимационные эффекты

После того, как показ одного изображения завершается, загружается следующее изображение. Когда его загрузка завершается, начинает воспроизводиться анимационный эффект, после чего загружается новое изображение. Моменты завершения визуализации и загрузки изображения идентифицируются с помощью соответствующих им событий, которые отслеживаются объектом `ImageViewer`.

##### ① Установка `ImageViewer`

Для `ImageViewer` устанавливается область размером в полный экран. Обработка событий завершения загрузки изображения и завершения процесса его отображения на экране выполняется в соответствующих функциях.

```

ImageViewer = new CImageViewer('Common ImageViewer');
ImageViewer.setFrameArea(0, 0, 960, 540);
ImageViewer.setOnBufferingComplete(function(){
    alert('Image ready!');
});
ImageViewer.setOnRenderingComplete(function(){
    alert('You can prepare next image!');
});
  
```

##### ② Установка режима слайд-шоу

Существует множество различий в последовательности воспроизведения изображений в обычном режиме и с использованием эффектов анимации. Для перехода в режим слайд-шоу необходимо вызвать функцию `startSlideshow()`, которая делает возможным применение этих эффектов.

```

ImageViewer.startSlideshow();
  
```

##### ③ Воспроизведение первого изображения

Функция `play()` инициирует воспроизведение первого изображения. Параметры, такие как ширина и высота, здесь роли не играют, т.к. в режиме слайд-шоу для изображений нет возможности указывать их местоположение.

```

ImageViewer.play('$WIDGET/picture_1.jpg', 640, 480);
  
```

##### ④ Подготовка следующего изображения к загрузке

Следующее изображение загружается после того, как было отображено первое. В момент, когда воспроизведение первого изображения завершено, начинает выполняться функция, которая была прописана в `setOnRenderingComplete()`. Далее вызывается функция `prepareNext()`, использующая в качестве параметров путь к изображению и анимационный эффект, который вы хотите воспроизвести. Затем происходит событие *Buffering Completed*, которое означает, что следующее изображение готово к показу.

```
ImageViewer.prepareNext('$WIDGET/picture_2.jpg ', ImageViewer.Effect.FADE2);
```

#### ⑤ Воспроизведение анимации

Воспроизведение анимационного эффекта начинается сразу после того, как была вызвана функция `showNow()`. Как только воспроизведение этого эффекта завершается, происходит событие *Rendering Complete*, означающее завершение процесса визуализации изображения. Следующий анимационный эффект воспроизводится путем повторения процесса, описанного в пункте ④. Необходимо помнить, что вызов функции `showNow()` должен быть выполнен после того, как произошло событие *Buffering Complete*.

#### ⑥ Выход из режима слайд-шоу

Выход из режима слайд-шоу осуществляется путем вызова функции `endSlideshow()`. После этого все изображения выводятся на экран в обычном режиме.

```
ImageViewer.endSlideshow();
```

### 5.6.4.3. Слайд-шоу и ограничения

Существуют несколько ограничений, касающихся использования анимационных эффектов.

- Масштабирование недоступно.

Т.к. анимационные эффекты затрагивают весь экран, одновременно с этим масштабирование изображений применено быть не может. Изображения должны воспроизводиться в полноэкранном режиме (960x540 пикселей).

```
setFrameArea(0, 0, 960, 540);
```

Изображения показываются в центре экрана в оригинальном размере. Если размер оригинального изображения превышает по ширине 1920 пикселей и по высоте 1080 пикселей, тогда изображение уменьшается с соответствующим коэффициентом пересчета и отображается в центре экрана.

- Во время воспроизведения анимации не выполняйте никаких других действий.

Если в момент воспроизведения анимации выполнять с изображениями какие-либо другие действия, это может привести к некоторым нарушениям. Рекомендуется выполнять другие действия с изображениями только после того, как произошло событие *Rendering complete*, которое обозначает, что воспроизведение анимационного эффекта завершено.

- Аппаратные ограничения

В некоторых моделях цифровых телевизоров SAMSUNG анимационные эффекты не поддерживаются, что обусловлено аппаратными ограничениями. Для проверки возможности использования анимации используйте функцию `CImageViewer.isEffectAvailable()`.

### 5.6.5. События

Объект `ImageViewer` позволяет контролировать ряд событий, возникающих в процессе показа изображения, которые вы можете обрабатывать в соответствующих функциях.

Событие	Описание	Функция
Начало буферизации	Это событие возникает в момент начала загрузки изображения из интернета сразу после успешного соединения с сервером.	<code>setOnBufferingStart()</code>
Завершение буферизации	Событие происходит, когда загрузка изображения завершена, и оно готово для воспроизведения.	<code>setOnBufferingComplete()</code>
Завершение визуализации	Событие происходит, когда изображение полностью воспроизведено на экране. В случае наложения анимационного эффекта событие возникает в момент завершения анимации.	<code>setOnRenderingComplete()</code>
Ошибка сети	Возникает в случае, когда невозможно получить доступ к URL изображения.	<code>setOnNetworkError()</code>
Ошибка визуализации	Событие возникает, если произошла ошибка при декодировании и визуализации изображения.	<code>setOnRenderError()</code>

### 5.6.6. Воспроизведение локальных файлов

Объект `ImageViewer` воспроизводит изображения, хранящиеся в приложении или на устройстве USB, а также изображения из интернета. Для доступа к изображениям, хранящимся в приложении или на USB-устройстве, необходимо указывать специальный путь, начинающийся с символа „\$“.

#### 5.6.6.1. Воспроизведение изображений, хранящихся в приложении

Переменная „\$WIDGET“ указывает на каталог, в котором содержится приложение. В функцию `ImageViewer.play()` в качестве параметра передается строка - путь к файлу изображения, как показано ниже.

```
ImageViewer.play('$WIDGET/picture_2.jpg ', 640, 480);
```

### 5.6.6.2. Воспроизведение изображений с USB

В пути к USB-устройству указывается переменная '\$USB\_DIR'. Чтобы иметь возможность воспроизводить файлы изображений с USB, необходимо определить, куда смонтировано USB-устройство, т.к. телевизор имеет несколько USB портов, и путь к USB будет зависеть от того, к какому порту оно подсоединено. Путь к устройству USB вычисляется с помощью плагина Storage. Для получения более подробной информации об этом плагине обратитесь к руководству Plugin guide. Здесь приводится пример, использующий только несколько из доступных функций.

- ① Как вычислить путь к устройству, смонтированному на USB.

Путь к USB-устройству определяем, используя плагин Storage. Если добавить к полученному пути строку „\$USB\_DIR“, получим путь к корневому каталогу USB-устройства.

```
// Указывается объект Plugin
var eStoragePlugin = document.getElementById('pluginObjectStorage');
// Подразумевается, что устройство USB подсоединено
var deviceId = eStoragePlugin.GetUSBDeviceID(0);
// Подразумевается, что устройство USB имеет один раздел
var mountPath = eStoragePlugin.GetUSBMountPath(deviceId, 0);
var USBRootPath = '$USB_DIR/' + mountPath; // путь к корневому каталогу USB
```

Ваш телевизор имеет более двух портов USB. Если вам необходимо получить доступ к нескольким USB-устройствам или к устройству USB с несколькими разделами, используйте функции плагина Storage.

- ② Как построить путь к файлу на устройстве USB

Прочитать каталог на устройстве USB можно, используя функцию readDir() из API браузера App Engine. Эта функция в качестве результата возвращает массив. Ниже приведен пример чтения данных из этого массива. Для получения более подробной информации см. раздел 8 File API.

```
var fileSystemObj = new FileSystem(); // App Engine API
var dirs = new Array();
var files = new Array();
var data = fileSystemObj.readDir(USBRootPath); // readDir() возвращает массив
for (var i = 0; i < data.length; i++) {
    if (data[i].isDir) {
        dirs.push(data[i].name);
    }
    else {
        files.push(data[i].name);
    }
}
```

Полный путь к файлу изображения на устройстве USB формируется по следующей схеме.

```
...
var USBImageFile = USBRootPath + '/' + directory + '/' + file;
```

В итоге получаем значение переменной USBImageFile равное:

```
'$USB_DIR/sda1/myphotos/picture_2.jpg'
```

- ③ Показ файла изображения, размещенного на устройстве USB.

```
ImageViewer.play(USBImageFile);
```

## 5.6.7. Справка

### 5.6.7.1. CImageViewer

CImageViewer	
Объект, отвечающий за воспроизведение изображений	
Путь к библиотеке	\$MANAGER_WIDGET/Common/API/CImageViewer.js
Синтаксис	New CImageViewer(objectName:String)
Параметры	objectName: Строка Включает имя создаваемого объекта. Это имя будет указано в отладочных сообщениях для ImageViewer.
Возвращаемое значение	CImageViewer object
Примечание	В действительности, ТВ-система содержит только один ImageViewer. Поэтому необходимо создавать в приложении один объект класса CImageViewer. Если для воспроизведения изображения используется ImageViewer, в момент завершения приложения должна быть вызвана функция destroy(). (Начиная с Samsung Smart TV version 2.269)
Пример	var ImageViewer = new CImageViewer('Common ImageViewer');
Атрибуты	□ Effect
Методы	□ clearScreen □ destroy □ endSlideshow □ getStopFlag □ hide □ isEffectAvailable □ isValidType □ play □ prepareNext □ setDisplayArea □ setFrameArea □ setOnBufferingComplete □ setOnBufferingStart □ setOnNetworkError □ setOnRenderError □ setOnRenderingComplete

	<ul style="list-style-type: none"> <li>▫ show</li> <li>▫ showNow</li> <li>▫ startSlideshow</li> <li>▫ stop</li> </ul>
--	---

#### 5.6.7.2. Effect

<b>Effect</b>	
Объект, содержащий значения для анимационных эффектов	
Путь к библиотеке	\$MANAGER_WIDGET/Common/API/CImageViewer.js
Синтаксис	<code>var ImageViewer = new CImageViewer(objectName:String); ImgeViewer.Effect.VALUE</code>
Параметры	None
Возвращаемое значение	None
Примечание	Не используйте метод INIT, который предназначен для внутренних целей. (Начиная с Samsung Smart TV version 2.269)
Пример	<code>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.Effect.FADE1;</code>
Атрибуты	<ul style="list-style-type: none"> <li>▫ NONE</li> <li>▫ FADE1</li> <li>▫ FADE2</li> <li>▫ BLIND</li> <li>▫ SPIRAL</li> <li>▫ CHECKER</li> <li>▫ LINEAR</li> <li>▫ STAIRS</li> <li>▫ WIPE</li> <li>▫ RANDOM</li> </ul>
Методы	None

#### 5.6.7.3. clearScreen()

<b>clearScreen</b>	
Удаляет изображения с экрана	
Синтаксис	<code>ImageViewer.clearScreen()</code>
Параметры	None
Возвращаемое значение	None
Примечание	Функция stop() не удаляет изображение, которое воспроизводится

	на экране. Чтобы удалить изображение, используйте функцию <code>clearScreen</code> . (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<code>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.clearScreen();</code>
<b>Enumeration</b>	None

**5.6.7.4. destroy()**

<b>destroy</b>	
Останавливает воспроизведение изображения и возвращает ImageViewer в исходное состояние	
<b>Синтаксис</b>	<code>ImageViewer.destroy()</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Если вы использовали ImageViewer, то данная функция должна быть вызвана, когда приложение завершается. Рекомендуется вызывать ее внутри функции, прописанной в свойстве <code>onunload</code> тэга <code>&lt;body&gt;</code> . (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<code>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.destroy ();</code>
<b>Enumeration</b>	None

**5.6.7.5. endSlideshow()**

<b>endSlideshow</b>	
Завершает воспроизведение изображений в режиме слайд-шоу	
<b>Синтаксис</b>	<code>ImageViewer.endSlideshow()</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Для завершения воспроизведения изображений в режиме слайд-шоу вызовите функцию <code>endSlideshow()</code> . Функции, такие как <code>prepareNext()</code> и <code>showNow()</code> , доступны только в режиме слайд-шоу. (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<code>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.startSlideshow(); ... // Use transition effect</code>

	<code>ImageViewer.endSlideshow();</code>
Enumeration	None

**5.6.7.6. `getStopFlag()`**

<b>getStopFlag</b>	
Проверяет рабочий статус <code>ImageViewer</code> .	
Синтаксис	<code>ImageViewer.getStopFlag();</code>
Параметры	None
Возвращаемое значение	Boolean true: Остановлен false : Работает
Примечание	(Начиная с Samsung Smart TV version 2.269)
Пример	<code>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.getStopFlag();</code>
Enumeration	None

**5.6.7.7. `hide()`**

<b>hide</b>	
Скрывает область, в которой воспроизводились изображения	
Синтаксис	<code>ImageViewer.hide();</code>
Параметры	None
Возвращаемое значение	None
Примечание	(Начиная с Samsung Smart TV version 2.269)
Пример	<code>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.show(); ... // Play images! ImageViewer.stop(); ImageViewer.hide();</code>
Enumeration	None

**5.6.7.8. *isEffectAvailable()***

<b>isEffectAvailable</b>	
Проверяет возможность использования анимационного эффекта	
<b>Синтаксис</b>	<code>ImageViewer.isEffectAvailable();</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	Boolean true: Пользовательская конфигурация позволяет использовать анимационные эффекты false : Эффекты не поддерживаются
<b>Примечание</b>	Некоторые модели не поддерживают анимационные эффекты, что обусловлено аппаратными ограничениями. В этих моделях в режиме слайд-шоу анимационные эффекты не работают. (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<code>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.isEffectAvailable();</code>
<b>Enumeration</b>	None

**5.6.7.9. *play()***

<b>Play</b>	
Воспроизводит изображения	
<b>Синтаксис</b>	<code>ImageViewer.play(url:String, width:Number, height:Number);</code>
<b>Параметры</b>	url: Строка Для воспроизведения доступны изображения, получаемые из интернета по URL, изображения, хранящиеся в приложении и на USB устройстве. width: Число Ширина указывается в пикселях. Изображение масштабируется и воспроизводится в области, выделенной в функции setFrameArea(). В режиме слайд-шоу этот параметр роли не играет. height: Число Высота изображения указывается в пикселях. Изображение масштабируется и воспроизводится в области, выделенной в функции setFrameArea().

	В режиме слайд-шоу этот параметр роли не играет.
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Воспроизводятся только изображения формата JPEG. (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setFrameArea(50,50,480,320); ImageViewer.play(image_url, image_width, image_height);</pre>
<b>Enumeration</b>	None

**5.6.7.10. prepareNext()**

<b>prepareNext</b>	
Иницирует загрузку следующего изображения с воспроизведением анимационного эффекта в режиме слайд-шоу	
<b>Синтаксис</b>	<code>ImageViewer.prepareNext(url:string, effect:ImageViewer.Effect);</code>
<b>Параметры</b>	url: Строка Для воспроизведения доступны изображения, получаемые из интернета по URL, изображения, хранящиеся в приложении и на USB устройстве. effect:ImageViewer.Effect Значение атрибута CImageViewer.Effect
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Вызов этой функции иницирует загрузку изображения. После того, как произошло событие <i>Buffering Complete</i> , функция <code>showNow()</code> начинает воспроизводить анимационный эффект. В некоторых моделях анимационные эффекты не поддерживаются, что обусловлено аппаратными ограничениями. Для проверки возможности использования анимации используйте функцию <code>CImageViewer.isEffectAvailable()</code> . (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setFrameArea(0,0,960,540); ImageViewer.startSlideshow(); ImageViewer.play(firstImage, 300, 400); // после события Rendering complete ImageViewer.prepareNext(NextImage, ImageViewer.Effect.FADE1); // после события Buffering complete ImageViewer.shwoNow(); ...</pre>

	// завершение слайд-шоу ImageViewer.endSlideshow();
Enumeration	None

#### 5.6.7.11. setFrameArea()

setFrameArea	
Задаёт область на экране, в которую помещаются изображения. Изображения воспроизводятся внутри заданной области.	
Синтаксис	ImageViewer.setFrameArea(left:Number, top:Number, width:Number, height:Number);
Параметры	left: Число Левая координата области top: Число Верхняя координата области width: Число Ширина области height: Число Высота области * в пикселях
Возвращаемое значение	None
Примечание	Если область задана следующим образом setFrameArea(0,0,225,200), то в медиа-источник передается верхняя левая зона экрана. Изображения до 450 x 400 пикселей воспроизводятся без изменений. Если размер изображения превышает стандартный, оно масштабируется в соответствии с пропорциями и отображается в заданной области. (Начиная с Samsung Smart TV version 2.269)
Пример	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setFrameArea(0,0,480,320); ImageViewer.play(image_url, image_width, image_height);</pre>
Enumeration	None

#### 5.6.7.12. setOnBufferingComplete()

setOnBufferingComplete	
Устанавливает функцию, которая будет вызываться в момент завершения процесса буферизации при загрузке изображения	
Синтаксис	ImageViewer.setOnBufferingComplete(callback:function);

Параметры	callback:function Функция, которая вызывается в момент завершения процесса буферизации
Возвращаемое значение	None
Примечание	Это событие происходит после завершения загрузки изображения. Если это событие произошло после вызова функции prepareNext(), это значит, что изображение готово к воспроизведению анимационного эффекта, который инициируется путем вызова функции showNow(). (Начиная с Samsung Smart TV version 2.269)
Пример	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setOnBufferingComplete(function(){ });</pre>
Enumeration	None

**5.6.7.13. setOnBufferingStart()**

<b>setOnBufferingStart</b>	
Устанавливает функцию, которая будет вызываться в момент начала процесса буферизации изображения	
Синтаксис	ImageViewer.setOnBufferingStart(callback:function);
Параметры	callback:function Функция, которая вызывается в момент начала процесса буферизации изображения
Возвращаемое значение	None
Примечание	Если изображение находится в интернете, то это событие происходит, когда после установления сетевого соединения начинается процесс загрузки изображения. (Начиная с Samsung Smart TV version 2.269)
Пример	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setOnBufferingStart(function(){ });</pre>
Enumeration	None

**5.6.7.14. setOnNetworkError()**

<b>setOnNetworkError</b>	
Устанавливает функцию, которая будет вызываться в случае возникновения ошибки сети	
Синтаксис	ImageViewer.setOnNetworkError(callback:function);

Параметры	callback:function Функция, которая вызывается в случае возникновения ошибки сети
Возвращаемое значение	None
Примечание	Это событие происходит, когда в процессе загрузки изображения из интернета происходит потеря сетевого соединения. (Начиная с Samsung Smart TV version 2.269)
Пример	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setOnNetworkError(function(){ });</pre>
Enumeration	None

**5.6.7.15. setOnRenderError()**

<b>setOnRenderError</b>	
Устанавливает функцию, которая будет вызываться в случае возникновения ошибки, произошедшей в момент декодирования или визуализации изображения	
Синтаксис	ImageViewer.setOnRenderError(callback:function);
Параметры	callback:function Функция, которая вызывается в случае ошибки, произошедшей в момент декодирования или визуализации изображения
Возвращаемое значение	None
Примечание	Это событие происходит, когда после буферизации возникает ошибка в процессе декодирования и визуализации изображения из-за того, что оно имеет формат, который не поддерживается. (Поддерживаются только файлы формата JPEG). (Начиная с Samsung Smart TV version 2.269)
Пример	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setOnRenderError(function(){ });</pre>
Enumeration	None

**5.6.7.16. setOnRenderingComplete ()**

<b>setOnRenderingComplete</b>	
Устанавливает функцию, которая будет вызываться в момент завершения процесса визуализации изображения	
Синтаксис	ImageViewer.setOnRenderingComplete(callback:function);
Параметры	callback:function Функция, которая вызывается в момент завершения процесса

	визуализации изображения
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	Это событие происходит, когда изображение полностью отображается на экране, а в случае использования анимационного эффекта - в момент завершения воспроизведения этого эффекта. (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setOnRenderingComplete(function(){ });</pre>
<b>Enumeration</b>	None

**5.6.7.17. show()**

<b>Show</b>	
Отображает на экране область, в которой будут воспроизводиться изображения	
<b>Синтаксис</b>	ImageViewer.show();
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	(Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.show(); ... // Play images! ImageViewer.stop(); ImageViewer.hide();</pre>
<b>Enumeration</b>	None

**5.6.7.18. showNow ()**

<b>showNow</b>	
Начинает воспроизведение анимационного эффекта. Загрузка изображения к этому моменту должна быть завершена.	
<b>Синтаксис</b>	ImageViewer.showNow();
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	В некоторых моделях анимационные эффекты не поддерживаются, что обусловлено аппаратными ограничениями. См. функцию isEffectAvailable(). Вызов функции prepareNext() начинает загрузку изображения. Функция showNow(), которая инициирует воспроизведение

	анимационного эффекта, должна быть вызвана в момент завершения процесса буферизации (событие Buffering complete). (Начиная с Samsung Smart TV version 2.269)
<b>Пример</b>	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setFrameArea(0,0,960,540); ImageViewer.startSlideshow(); ImageViewer.play(firstImage, 300, 400); // после события Rendering complete ImageViewer.prepareNext(NextImage, ImageViewer.Effect.FADE1); // after Buffering complete event ImageViewer.shwoNow(); ... // завершение слайд-шоу ImageViewer.endSlideshow();</pre>
<b>Enumeration</b>	None

**5.6.7.19. startSlideshow ()**

<b>startSlideshow</b>	
Запускает воспроизведение изображений в режиме слайд-шоу	
<b>Синтаксис</b>	ImageViewer.startSlideshow()
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	<p>Вызовите функцию startSlideshow() для входа в режим слайд-шоу с эффектами анимации.</p> <p>Такие функции, как prepareNext() и showNow(), доступны только в режиме слайд-шоу.</p> <p>(Начиная с Samsung Smart TV version 2.269)</p>
<b>Пример</b>	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.startSlideshow(); ... // Use transition effect ImageViewer.endSlideshow();</pre>
<b>Enumeration</b>	None

**5.6.7.20. stop ()**

<b>stop</b>
Останавливает обработку изображений

<b>Синтаксис</b>	<code>ImageViewer.stop();</code>
<b>Параметры</b>	None
<b>Возвращаемое значение</b>	None
<b>Примечание</b>	<p>Эта функция не останавливает воспроизведение изображения на экране. Она останавливает такие процессы, как загрузка изображения и анимационные эффекты. Для удаления изображения с экрана используйте функцию <code>clearScreen()</code>.</p> <p>(Начиная с Samsung Smart TV version 2.269)</p>
<b>Пример</b>	<pre>var ImageViewer = new CImageViewer('Common ImageViewer'); ImageViewer.setFrameArea(50,50,480,320); ImageViewer.play(image_url, image_width, image_height); ImageViewer.stop();</pre>
<b>Enumeration</b>	None

## 6. Рекомендации для разработчиков приложений

Этот раздел содержит информацию, которая будет полезна разработчикам приложений.

### 6.1. Функция, которая должна быть вызвана при открытии приложения.

Для того чтобы приложение можно было запустить, оно должно быть отображено на экране. Для этого приложение должно дать соответствующую команду Менеджеру приложений. Только тогда, когда Менеджер приложений получит эту команду, он отобразит приложение на экране. Любой запрос к Менеджеру приложений делается путем вызова функций т.н. общего модуля (библиотеки). Используйте функцию `sendReadyEvent()` библиотеки `Widget`.

```
var widgetAPI = new Common.API.Widget(); // Создание объекта общего модуля
widgetAPI.sendReadyEvent();              // Сообщение Менеджеру приложений о готовности
```

### 6.2. Точка входа

Каждый раз, когда вы открываете приложение, браузер App Engine читает документ `index.html`. Если вы выбираете из списка нужное вам приложение и нажимаете кнопку `Enter`, браузер начинает чтение файла `index.html`. В этом файле в свойстве `onload` тэга `<body>` указывается имя функции JavaScript, которая в момент загрузки страницы запускает приложение. В первую очередь в приложении должна быть вызвана функция `sendReadyEvent()` объекта общего модуля `Widget`.

```
<body onload="Main.onLoad();">
```

### 6.3. Обработка событий, соответствующих нажатию кнопок на пульте управления

При нажатии любой кнопки на пульте управления происходит событие `"keydown"`. В этот момент управление передается активному (в фокусе) элементу экрана. Если в свойстве `"onkeydown"` этого элемента прописана функция, то она выполняется. Эта функция определяет, какая кнопка была нажата, используя свойство `event.keyCode`.

Вы должны помнить, что на нажатие кнопки может реагировать только тот элемент, который может стать активным на экране (может получить т.н. фокус). Если такого элемента нет, на экране телевизора не будет видно никакой реакции, какую бы кнопку вы не нажали.

Для всех кнопок пульта управления определены соответствующие им переменные объекта `tvKeyValue`.

Добавим в файл `index.html` элемент, который может получать фокус на экране.

```
<a href='javascript:void(0);' id='anchor' onkeydown='Main.keyDown();'></a>
```

Создадим функцию, которая будет обрабатывать событие `"onkeydown"`.

```
Main.keyDown = function(){                                // Обработчик нажатия кнопки
    var keyCode = event.keyCode;

    alert("Main Key code : " + keyCode);
    switch (keyCode) {
        case tvKey.KEY_LEFT:
            /**
```

```

        * Code for Left key event!
        */
        break;
    case tvKey.KEY_RIGHT:
        break;
    case tvKey.KEY_UP:
        break;
    case tvKey.KEY_DOWN:
        break;
    case tvKey.KEY_ENTER:
        break;
    case tvKey.KEY_RETURN:
        break;
    }
}

```

## 6.4. Кнопки пульта управления

Для каждого приложения Менеджер приложений регистрирует базовый набор кнопок, ему соответствующих. При нажатии зарегистрированной таким образом кнопки, ее событие (key event) загружается в приложение. Для незарегистрированных кнопок key event в приложение не загружается. Если нажата незарегистрированная кнопка, TV система закрывает приложение, и становятся доступными функции телевизора. Например, если в момент выполнения приложения, в котором не зарегистрирована кнопка управления громкостью, нажать эту кнопку, то работающее приложение будет принудительно закрыто, и на экране появится баннер громкости.

Наборы кнопок, которые Менеджер приложений регистрирует для приложений, приведены ниже. Если в файле config.xml значение тэга <fullwidget> равно “y” (полноэкранное приложение) и при этом значение Full-screen Application keyset равно “n”, то регистрируется набор, соответствующий оконному (Single-wide) приложению.

Полноэкранное приложение		Оконное приложение
VOL_UP	KEY_1	KEY_WHEELDOWN
KEY_VOL_DOWN	KEY_2	KEY_WHEELUP
KEY_MUTE	KEY_3	KEY_RED
KEY_TOOLS	KEY_4	KEY_GREEN
KEY_INFO	KEY_5	KEY_YELLOW
KEY_EMODE	KEY_6	KEY_BLUE
KEY_DMA	KEY_7	KEY_RW
KEY_MENU	KEY_8	KEY_PAUSE
KEY_SOURCE	KEY_9	KEY_FF
KEY_PRECH	KEY_0	KEY_PLAY
KEY_FAVCH	KEY_WHEELDOWN	KEY_STOP
KEY_CHLIST	KEY_WHEELUP	KEY_ENTER
KEY_DMA	KEY_RED	KEY_RETURN
KEY_TTX_MIX	KEY_GREEN	KEY_EXIT
KEY_GUIDE	KEY_YELLOW	
KEY_SUBTITLE	KEY_BLUE	
KEY_ASPECT	KEY_RW	
KEY_DOLBY_SRR	KEY_PAUSE	
KEY_MTS	KEY_FF	
KEY_PANEL_CH_UP	KEY_PLAY	
KEY_PANEL_CH_DOWN	KEY_STOP	
KEY_PANEL_VOL_UP	KEY_ENTER	
KEY_PANEL_VOL_DOWN	KEY_RETURN	

KEY_PANEL_ENTER KEY_PANEL_SOURCE KEY_PANEL_MENU	KEY_EXIT	
---	----------	--

В любом приложении можно дополнительно с помощью функции `registKey()` общего модуля „Plugin“ регистрировать/отключать необходимые кнопки. Необходимо обратить внимание:

- Регистрация/отключение кнопок должно происходить после события `window.onshow`.

Кнопки, зарегистрированные/отключенные после события `window.onshow` обрабатываются корректно, если же они были зарегистрированы до этого события, то они будут игнорироваться. Событие `window.onshow` происходит после вызова функции `sendReadyEvent()` общего модуля „Widget“. Ниже приведен пример.

```
var WIDGET = new Common.API.Widget(); // Для sendReadyEvent()
var TVKEY = new Common.API.TVKeyValue(); // Объект, содержащий значения кнопок пульта
var PLUGIN = new Common.API.Plugin(); // Общий модуль Plugin

Main.onLoad = function(){

    window.onshow = function(){ // Функция, выполняющаяся по событию onshow

        PLUGIN.registKey(TVKEY.KEY_VOL_UP);
        PLUGIN.registKey(TVKEY.KEY_VOL_DOWN);

    }

    WIDGET.sendReadyEvent();
}
/**
 * code
 */
```

После события `window.onshow` момент регистрации кнопок не имеет значения.

## 6.5. Завершение приложения

Существуют два способа закрытия работающего приложения: с помощью нажатия кнопки Return или Exit на пульте управления или посредством вызова события, которое будет обработано Менеджером приложений. Для этого используются функции `sendExitEvent()` и `sendReturnEvent()` общего модуля Widget. Для получения более детальной информации см. раздел 5 “Общие модули”.

## 6.6. Предотвращение закрытия приложения при нажатии кнопок Return или Exit.

Используя кнопки Return или Exit на пульте управления, вы можете закрыть приложение. Если вы нажимаете кнопку Return, вы переходите к Менеджеру приложений, если же вы нажали кнопку Exit, вы возвращаетесь на экран телевизора. Чтобы предотвратить выход из приложения при нажатии одной из этих кнопок, нужно использовать функцию `blockNavigation()` общего модуля Widget. Для получения более детальной информации см. раздел 5 “Общие модули”.

## 6.7. Дополнительная информация

В момент выполнения приложения Менеджер приложений может предоставлять вам дополнительную информацию, которую приложение получает по ссылке `window.location.search`, как

показано ниже. Таким образом, в приложении можно получить введенные через SSO ID и пароль, а также информацию об установленных стране и языке телевизора.

```
?country=US&language=1&modelid=Valencia&server=development&remocon=2_35_259&area=USA&product=1&mgrver=3.000&id=ori590e@exxx.com&pw=pxxxx&payload={key1:value2,key2:value2}
```

Атрибут	Значение	Описание
country	Код страны	Код страны. См. раздел 9.1.
language	Код языка	<i>Заменен атрибутом 'lang'</i>
modelid	ID модели цифрового телевизора	ID модели цифрового телевизора
server	development   operating	Тип сервера: сервер разработки или рабочий
remocon	-	Не используется
area	Код региона	Код региона. См. раздел 9.3.
Id	Строка	Id учетной записи, зарегистрированной в SSO. Передается только в случае, когда приложение запущено после входа в ТВ-аккаунт (после ввода пользователем логина и пароля). Перед использованием должна быть вызвана функция decodeURIComponent().
pw	Строка	Пароль, зарегистрированный в SSO. Передается только в случае, когда приложение запущено после входа в ТВ-аккаунт. Перед использованием должна быть вызвана функция decodeURIComponent().
product	Категория продукта	См. раздел 9.4. Категория продукта
mgrver	Версия Samsung Smart TV	Если для работы приложения требуется специфическая версия Samsung Smart TV, вы должны проверять значение mgrver.
payload	Строка(JSON)	Дополнительная информация для специальной услуги (биллинг). Передается только в случае, когда приложение запущено после отправки уведомления.

## 6.8. Что должен содержать CSS

Элемент, выходящий за границы, обусловленные максимально допустимым разрешением телевизора (960 x 540 pixel), может привести к необходимости прокручивания содержимого экрана. В файле index.html границы всех элементов (margins and paddings) должны быть по умолчанию равны 0 для того, чтобы избежать выхода за границы экрана. Ваш CSS должен содержать следующие строки.

```
body {
    margin: 0;
```

```
padding: 0;
background-color: transparent;
}
```

## 6.9. Предварительная загрузка изображений

Браузер App Engine декодирует изображения для того, чтобы они могли нормально отображаться на вашем экране. Может случиться, что при прокрутке изображений с большой скоростью, процесс декодирования будет тормозить их отображение. Для того чтобы этого избежать, вы можете загружать и сохранять изображения до того, как вы будете их использовать. Таким образом, вы экономите время на декодировании изображений.

```
var imagePreloading = new Image();
imagePreloading.src = "Resource/image/largeNFastChange.png";
```

С помощью приведенного выше кода вы можете декодировать изображения. Один раз загруженное в браузер изображение не нуждается в декодировании еще раз, это позволит вашему телевизору отображать его быстрее.

## 6.10. Особенности тэга <input>

Объект, созданный с помощью тэга <input>, не перекрывается другими объектами. Даже если вы используете z-index, объект <input> всегда будет находиться поверх других объектов.

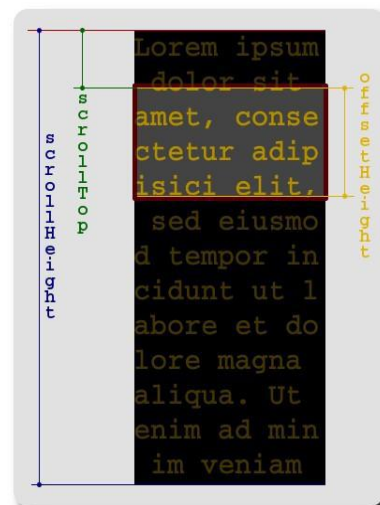
## 6.11. Полоса прокрутки

Браузер App Engine поддерживает функцию прокрутки, однако, вы не можете увидеть на экране полосу прокрутки. Если содержимое экрана не умещается в пределах области, определенной для него заранее, прокручивать его вверх и вниз вы можете с помощью кнопок на пульте управления. Если вы хотите увидеть актуальную полосу прокрутки на экране телевизора, для того чтобы поместить ее на экран, используйте функции `scrollHeight`, `scrollTop` и `offsetHeight`.

Поместите якорь („<a>“) в том месте, которое вы намерены прокручивать, переведите на него фокус и нажмите на пульте управления кнопку вверх или вниз, что позволит вам прокручивать содержимое экрана.

(Picture Source:

<http://sevencapitalsins.wordpress.com/2007/07/04/javascript-smooth-scroll/>)



## 6.12. Создание динамических веб-страниц.

Если вы хотите создать динамическую веб-страницу, вы можете использовать функцию `createElement` и свойство `innerHTML`. Функция `createElement` - это стандартный метод работы с объектом HTML DOM, его недостатком является низкая оперативная скорость по сравнению с `innerHTML`. Рекомендуется использовать свойство `innerHTML`, т.к. ваше приложение выполняется в телевизоре, который предоставляет менее оптимизированную рабочую среду, чем персональный компьютер.

### 6.13. Значение для межбуквенного расстояния

Значение по умолчанию для межбуквенного расстояния в браузере APP Engine равно -1. В случае с тайским языком это приводит к наложению букв одна на другую. Чтобы этого избежать, установите значение равным 0. Включите следующий код в ваш файл CSS.

```
* {  
letter-spacing: 0px;  
}
```

### 6.14. Управление памятью

Если в своем приложении вы динамически изменяете код страницы с помощью свойства innerHTML и создаете несколько экземпляров объекта XMLHttpRequest, то это может привести к переполнению памяти вашего телевизора, и установленные на нем приложения не будут работать должным образом. Несмотря на наличие т.н. сборщика мусора, эта проблема остается актуальной. Кроме того, не стоит забывать, что по сравнению с персональным компьютером, ресурсы памяти телевизора ограничены. Поэтому, если вы хотите написать “оптимальное” приложение, которое будет работать “корректно”, вы должны предусмотреть дополнительные меры для управления памятью.

#### 6.14.1. Использование XMLHttpRequest

Если вы в своем приложении часто создаете и используете XHR объект, вы можете столкнуться с проблемой некорректной работы вашего приложения. Решение этой проблемы предлагается в разделе 4.6. Пример взаимодействия с объектом XMLHttpRequest.

#### 6.14.2. Освобождение памяти

Если содержимое тэга заменяется с помощью функции innerHTML, соответствующий узел объекта DOM закрывается, затем на его месте создается и подсоединяется новый узел. Закрытый узел остается в памяти. Если это происходит часто, то вы можете столкнуться с проблемой переполнения памяти. Решить эту проблему помогает использование функции putInnerHTML() общего модуля (библиотеки) Widget.

### 6.15. Watch Dog

Система управления телевизора SAMSUNG периодически проверяет, корректно ли работает приложение. Если приложение остается недоступным в течение 20 секунд, оно принудительно закрывается. Эта система получила название “Watch dog”. Включить и выключить Watch dog в приложении можно с помощью функций setOnWatchDog(), setOffWatchDog() общего модуля „Plugin”. В приложении сетевые операции и операции, связанные с обработкой большого массива данных, могут выполняться более 20 секунд, и возникает риск, что они будут принудительно прерваны системой Watch dog. Поэтому перед выполнением таких операций рекомендуется выключать Watch dog. И после их завершения включать Watch dog снова. Если вы оставите Watch dog выключенным, ваш телевизор может перестать работать.

### 6.16. Предотвращение искажения экранных меню OSD (ON Screen Display)

Изображения, используемые в приложении, могут исказиться, если в фоновом режиме на экране прокручивается видеоряд с быстрой модуляцией (например, идет какой-либо фильм с динамичным сюжетом lol ). Использование функций setOnOSDState(), setOffOSDState() общего модуля „Plugin” помогает решить эту проблему. Для получения более детальной информации по их использованию смотрите раздел 5.5. Plugin. Эта проблема может возникнуть в оконных (Single-wide) и Ticker-приложениях (но не в полноэкранных приложениях).

## 6.17. Поддержка переполнения текста text-overflow в CSS

Браузер APP Engine поддерживает стандарты CSS1 и CSS2. Кроме этого поддерживается атрибут текстового переполнения text-overflow, являющийся элементом стандарта CSS3. Таким образом, браузер может справляться с проблемой текстового переполнения более эффективно.

## 6.18. Инструкции по созданию видео-приложений.

Видео-приложения, использующие плеер (Plugin), могут быть закрыты только после его остановки. Если вы завершите приложение без остановки плеера, может возникнуть критическая проблема, такая как приостановка работы телевизора (пауза). Лучшим способом предотвращения такой проблемы является остановка плеера в функции, прописанной в свойстве onunload тэга <body>.

## 6.19. Использование IME

С помощью пульта управления в приложении можно вводить текст. Samsung Smart TV поддерживает модуль, позволяющий использовать редактор IME (Input Method Editor). Этот модуль дает пользователям возможность вводить языковые символы. Для получения детальной информации, обратитесь к документации IME Tutorial manual, которая предоставляется отдельно.

## 6.20. Оптимизация скорости первоначальной загрузки приложения

### 6.20.1. Динамическая загрузка Javascript.

Вы можете уменьшить время загрузки приложения, минимизировав количество кода Javascript на начальной странице. Вместо того чтобы загружать весь код Javascript на начальной странице, его можно подгружать динамически по мере необходимости. Разделите ваш большой Javascript файл на несколько файлов. Для загрузки начальной страницы приложения вам понадобятся только некоторые из них, их число вы можете минимизировать. Остальные файлы вы сможете подключать динамически, как показано ниже.

```
Main.onLoad = function(){
    var widgetAPI = new Common.API.Widget();
    widgetAPI.sendReadyEvent();
    setTimeout('IncludeJavascript()',0);
}
function IncludeJavascript(){
    var head= document.getElementsByTagName('head')[0];
    var script = document.createElement('script'); //создание тэга script
    script.type = 'text/javascript';
    script.src = 'test.js';
    script.onload = function () {...};
    head.appendChild(script); //добавление нового элемента script на страницу
}
```

### 6.20.2. Динамическая загрузка фоновых изображений CSS

Браузер App Engine загружает фоновое изображение для элемента страницы в тот момент, когда к нему применяется стиль, прописанный в CSS. Вы можете повысить скорость загрузки приложения, если для элементов, которые не нужны на начальной странице, стиль фона будет применен динамически, когда это необходимо, как показано в следующем примере.

```
function loadBackgroundImage(eid, url){
    document.getElementById(eid).style.backgroundImage = "url("+url+")";
}
function loadCSSImage(){
    loadBackgroundImage("Full1", "img/Full_1.png");
    loadBackgroundImage("Full2", "img/Full_2.png");
    loadBackgroundImage("Full3", "img/Full_3.png");
}
```

### 6.20.3. Разделение функций Main.onLoad

Если вы будете корректно использовать функцию `setTimeout`, начальная страница приложения сможет отображаться быстрее. Разделите функцию `Main.onLoad` на два этапа. Первый будет отвечать за формирование внешнего вида начальной страницы (загрузку фоновых изображений). Второй будет включать остальные действия, которые не отвечают за внешний вид начальной страницы. Следующий пример иллюстрирует эту технику.

```
Main.onLoad = function(){
    ...generate background

    var widgetAPI = new Common.API.Widget();
    widgetAPI.sendReadyEvent();
    setTimeout('Main_onLoad2()',0);
}
function Main_onload2(){
    IncludeJavascript();    // см. 6.20.1
    generateDetailPage();
}
```

## 6.21. Инструкции по динамической загрузке стилей CSS

Динамическая загрузка файлов стилей CSS может быть выполнена путем простого добавления узла `link` к тэгу `<head>`. Однако, CSS должен быть загружен до того, как браузер App Engine построит дерево HTML DOM. Только тогда стили CSS смогут быть применены к элементам HTML. Если CSS загружается после конфигурирования DOM, тогда к объекту DOM стили не применяются. Посмотрите пример, приведенный ниже.

- ① Добавляем элементы HTML после динамической загрузки CCS.

Загружаем файл `red_box.css`, и затем с задержкой в 3 секунды добавляем элемент `div`. Запустив приложение, вы увидите, что через 3 секунды на экране появится красный квадрат, потому что CSS файл был загружен раньше, чем DOM.

```
var Main = {
```

```

}
var WIDGET = new Common.API.Widget(); // For sendReadyEvent()
Main.onLoad = function() {          // The first function

    alert("Main : onLoad()");
    WIDGET.sendReadyEvent();
    includeCSS ('red_box.css');
    setTimeout("insertHTML()", 3000);
//    insertHTML();
//    setTimeout("includeCSS ('red_box.css')", 3000);
}
function includeCSS(path) {

    alert('++++++ includeCSS ++++++');
    var linkNode = document.createElement('link');
    linkNode.type = 'text/css';
    linkNode.rel = 'stylesheet';
    linkNode.href = path;
    document.getElementsByTagName('head')[0].appendChild(linkNode);
}
function insertHTML() {

    alert('++++++ insertHTML ++++++');
    var divNode = document.createElement('div');
    divNode.id = 'red_box';
    document.body.appendChild(divNode);
}
* red_box.css

#red_box {

    position : absolute;
    left: 100px;
    top: 100px;
    width: 100px;
    height: 100px;
    border : 3px solid #FF0000;
}

```

## ② Добавляем элементы HTML и затем загружаем CCS

В приведенном выше примере в функции Main.onLoad() удалите символы комментария и закомментируйте предыдущие две строки. Теперь CSS загружается через 3 секунды после добавления элемента HTML Div.

Теперь никаких изменений на экране не произойдет, и красный квадрат вы не увидите.

```

//    includeCSS ('red_box.css');
//    setTimeout("insertHTML()", 3000);
    insertHTML();
    setTimeout("includeCSS ('red_box.css')", 3000);

insertHTML();
setTimeout("includeCSS ('red_box.css')", 3000);

```

## 7. Браузер App Engine

Этот раздел посвящен браузеру App Engine, в котором выполняются приложения Samsung Smart TV.

### 7.1. Что собой представляет App Engine

- Stands for Markup engine Platform for Embedded Systems
- Browser engine for Consumer Electronics (CE) devices
- Controls devices using markup tags and script classes

В телевизорах Samsung с функцией Samsung Smart TV приложения отображаются и выполняются в браузере App Engine. В то время как Internet Explorer и Firefox являются браузерами для персонального компьютера, App Engine является браузером для телевизоров Samsung.

### 7.2. Технические характеристики

- **Поддерживаемые веб-стандарты**  
HTML4.01, XHTML1.0, XML1.0 Markup language specifications  
HTTP1.0/1.1  
CSS1, CSS2, CSS TV Profile 1.0  
DOM1, DOM2  
JavaScript 1.6
- **Поддерживаемые форматы изображений**  
BMP, JPEG, PNG, GIF

## 8. Файловый программный интерфейс приложения (File API)

Этот раздел содержит информацию о системе файлового ввода/вывода, которую поддерживает браузер Samsung.

### 8.1. Что такое File API

Приложения могут использовать систему файлового ввода/вывода, которую поддерживает браузер App Engine.

Приложения могут сохранять данные, которые они получают в процессе работы, и читать их в любой момент, когда в этом возникнет необходимость. Все приложения выполняют операции ввода/вывода в общей для всех приложений области памяти. Файлы, используемые в разных приложениях, могут иметь одинаковые имена. Поэтому рекомендуется создавать для каждого приложения свой каталог, в имени которого содержится ID приложения (curWidget.id), и работать с файлами приложения в этом каталоге.

Если несколько приложений используют общую информацию, они должны указывать путь к общим файлам в соответствии с установленной заранее договоренностью.

Операции файлового ввода/вывода осуществляются путем создания объекта файловой системы и вызова функций (интерфейсов) этого объекта.

### 8.2. Описание API

Ниже приводится список функций (интерфейсов), которые поддерживает браузер Samsung. Вы можете вызывать их в коде JavaScript.

### 8.2.1. FileSystem()

FileSystem	
Создает экземпляр объекта файловой системы для работы с файлами в приложении	
Синтаксис	<code>new FileSystem();</code>
Параметры	None
Возвращаемое значение	App Engine FileSystem object
Примечание	Возвращает экземпляр объекта файловой системы браузера App Engine, благодаря чему становятся доступными чтение и запись в текстовые файлы. Бинарный формат файлов не поддерживается.
Пример	<code>var fileSystemObj = new FileSystem();</code>
Атрибуты	□□None
Методы	<ul style="list-style-type: none"> <li>▪ <code>openCommonFile</code></li> <li>▪ <code>closeCommonFile</code></li> <li>▪ <code>deleteCommonFile</code></li> <li>▪ <code>createCommonDir</code></li> <li>▪ <code>deleteCommonDir</code></li> <li>▪ <code>isValidCommonPath</code></li> <li>▪ <code>readLine</code></li> <li>▪ <code>writeLine</code></li> <li>▪ <code>readAll</code></li> <li>▪ <code>writeAll</code></li> <li>▪ <code>readDir</code></li> <li>▪ <code>openFile</code></li> </ul>

### 8.2.2. openCommonFile()

openCommonFile	
Открывает файл в общей области	
Синтаксис	<code>fileSystemObj.openCommonFile(filePath:String, mode:String)</code>
Параметры	<p><code>filePath</code>: Строка путь, включающий имя файла.</p> <p><code>mode</code>: Строка</p> <p><code>r</code>: Открыть файл для чтения. Файл должен существовать.</p> <p><code>w</code>: Создать пустой файл для записи. Если файл с таким именем уже существует, содержимое файла удаляется и файл обрабатывается как новый пустой файл.</p> <p><code>a</code>: Запись в файл. Данные добавляются в конец файла. Если файл по указанному пути не найден, создается новый файл.</p> <p><code>r+</code>: Открыть файла для изменений: чтение/запись. Файл должен существовать.</p> <p><code>w+</code>: Создать новый пустой файл для чтения и записи. Если файл с таким именем уже существует, содержимое файла удаляется и файл</p>

	обрабатывается как новый пустой файл. <b>a+:</b> Открыть файл для чтения и записи. Данные добавляются в конец файла, при этом данные, содержащиеся в файле, защищены от перезаписи. Вы можете установить внутренний курсор в файле на заданную позицию (используя <code>fseek</code> , <code>rewind</code> ), но при записи курсор переместится в конец файла. Если файл по указанному пути не найден, создается новый файл.
<b>Возвращаемое значение</b>	App Engine file object
<b>Примечания</b>	С помощью этой функции приложения могут открывать с файлы для чтения/записи в общей для всех приложений области памяти. Так как файлы, используемые в разных приложениях, могут иметь одинаковые имена, рекомендуется создавать для каждого приложения свой каталог, в имени которого содержится ID приложения ( <code>curWidget.id</code> ), и работать с файлами приложения в этом каталоге.
<b>Пример</b>	<pre>var fileSystemObj = new FileSystem(); var fileObj = fileSystemObj.openCommonFile(curWidget.id + '/testFile.data', 'w'); fileObj.writeAll('something to write.');</pre> <pre>fileSystemObj.closeCommonFile(fileObj);</pre>
<b>Enumeration</b>	None

### 8.2.3. `closeCommonFile()`

<b>closeCommonFile</b>	
Закрывает файл, открытый с помощью функции <code>openCommonFile()</code> .	
<b>Синтаксис</b>	<code>fileSystemObj.closeCommonFile(fileObj:App Engine file object)</code>
<b>Параметры</b>	<p><code>fileObj</code>: App Engine file object</p> <p>Экземпляр файлового объекта, созданный с помощью функции <code>openCommonFile()</code></p>
<b>Возвращаемое значение</b>	Boolean
<b>Примечания</b>	Открытые файлы должны быть закрыты. Это особенно актуально в случае, когда запись в файл осуществлялась с помощью функции <code>writeAll</code> , иначе результат записи в файл не будет сохранен.
<b>Пример</b>	<pre>var fileSystemObj = new FileSystem(); var fileObj = fileSystemObj.openCommonFile(curWidget.id + '/testFile.data', 'w'); fileObj.writeAll('something to write.');</pre> <pre>fileSystemObj.closeCommonFile(fileObj);</pre>
<b>Enumeration</b>	None

### 8.2.4. `deleteCommonFile()`

**deleteCommonFile**

Удаляет файл из файловой системы.

Синтаксис	fileSystemObj.deleteCommonFile(filePath:String)
Параметры	filePath: Строка путь к файлу, который требуется удалить
Возвращаемое значение	Boolean
Примечания	None
Пример	<pre>var fileSystemObj = new FileSystem(); var bResult = fileSystemObj.deleteCommonFile(curWidget.id + '/testFile.data');</pre>
Enumeration	None

**8.2.5. createCommonDir()****createCommonDir**

Создает каталог в файловой системе.

Синтаксис	fileSystemObj.createCommonDir(directoryPath:String)
Параметры	directoryPath: Строка путь к каталогу, который требуется создать
Возвращаемое значение	Boolean
Примечания	Создание для каждого приложения каталога, содержащего в названии идентификатор приложения (curWidget.id), позволит избежать ошибок, связанных с именованием файлов в приложениях.
Пример	<pre>var fileSystemObj = new FileSystem(); var bResult = fileSystemObj.createCommonDir(curWidget.id);</pre>
Enumeration	None

**8.2.6. deleteCommonDir()****deleteCommonDir**

Удаляет каталог из файловой системы.

Синтаксис	fileSystemObj.deleteCommonDir(directoryPath:String)
-----------	---

Параметры	directoryPath: Строка путь к каталогу, который требуется удалить
Возвращаемое значение	Boolean
Примечания	None
Пример	<pre>var fileSystemObj = new FileSystem(); var bResult = fileSystemObj.deleteCommonDir(curWidget.id);</pre>
Enumeration	None

### 8.2.7. isValidCommonPath()

<b>sValidCommonPath</b>	
Проверяет, существует ли данный каталог	
Синтаксис	fileSystemObj.isValidCommonPath(directoryPath:String)
Параметры	directoryPath: Строка путь к каталогу, существование которого необходимо проверить
Возвращаемое значение	Целое число 0: ошибка внутри JavaScript функции 1: каталог существует 2: каталог не существует
Примечания	None
Пример	<pre>var fileSystemObj = new FileSystem(); var bValid = fileSystemObj.isValidCommonPath(curWidget.id); if (!bValid) {     fileSystemObj.createCommonDir(curWidget.id); }</pre>
Enumeration	None

### 8.2.8. readLine()

<b>readLine</b>	
Читает строку в открытом файле.	
Синтаксис	fileObj.readLine()
Параметры	None
Возвращаемое значение	Строка Возвращает строку до символа перевода строки. В случае отсутствия строки для чтения возвращает null.

Примечания	None
Пример	<pre>var fileSystemObj = new FileSystem(); var fileObj = fileSystemObj.openCommonFile(curWidget.id + '/testFile.data', 'r'); var strLine = ""; var arrResult = new Array(); while((strLine=fileObj.readLine())) { arrResult.push(strLine); }</pre>
Enumeration	None

### 8.2.9. writeLine()

<b>writeLine</b>	
Записывает строку в открытый файл.	
Синтаксис	fileObj.writeLine(text:String)
Параметры	text:String строка, которую требуется записать в файл
Возвращаемое значение	Boolean
Примечания	None
Пример	<pre>var fileSystemObj = new FileSystem(); var fileObj = fileSystemObj.openCommonFile(curWidget.id + '/testFile.data', 'w'); fileObj.writeLine('something to write.');</pre>
Enumeration	None

### 8.2.10. readAll()

<b>readAll</b>	
Читает все содержимое открытого файла.	
Синтаксис	fileObj.readAll()
Параметры	None
Возвращаемое значение	Строка Все содержимое файла
Примечания	None
Пример	<pre>var fileSystemObj = new FileSystem();</pre>

	<pre>var fileObj = fileSystemObj.openCommonFile(curWidget.id + '/testFile.data', 'r'); var strResult = fileObj.readAll(); alert(strResult);</pre>
Enumeration	None

### 8.2.11. writeAll()

<b>writeAll</b>	
Записывает несколько строк в открытый файл.	
Синтаксис	fileObj.writeAll(text:String)
Параметры	text:String текст, который требуется записать в файл (может содержать символы перевода на новую строку)
Возвращаемое значение	Boolean
Примечания	None
Пример	<pre>var fileSystemObj = new FileSystem(); var fileObj = fileSystemObj.openCommonFile(curWidget.id + '/testFile.data', 'w'); fileObj.writeAll('something to write.');</pre> fileSystemObj.closeCommonFile(fileObj);
Enumeration	None

### 8.2.12. readDir()

<b>readDir</b>	
Читает каталог на USB устройстве.	
Синтаксис	fileSystemObj.readDir(directoryPath:String)
Параметры	directoryPath:String Путь к каталогу на USB
Возвращаемое значение	Array Массив, содержащий информацию о файлах в каталоге. Можно получить следующие данные: name : имя файла/каталога isDir : Если элемент массива - каталог, то isDir = true, иначе false size : размер файла (в байтах) atime : Последнее время открытия файла / последнее время обращения к каталогу через команду cd mtime : Время последней модификации содержимого файла ctime : Время последней модификации информации о файле

	Если по указанному пути файл или каталог не найдены, функция возвращает значение false.
Примечания	Для доступа к данным на устройстве USB используйте переменную „\$USB_DIR“. См. Storage plugin.
Пример	<pre>var fileSystemObj = new FileSystem(); var usbPath = '\$USB_DIR' + usb_mount_path; var arrFiles = fileSystemObj.readDir(usbPath); if (arrFiles) {   for (var i=0; i &lt; arrFiles.length; i++) {     alert(arrFiles[i].name);     alert(arrFiles[i].isDir);   } }</pre>
Enumeration	None

### 8.2.13. openFile()

<b>openFile</b>	
Открывает файл в приложении.	
Синтаксис	fileSystemObj.openFile(filePath:String, mode:String)
Параметры	<p>filePath:Строка путь к файлу, включая имя</p> <p>mode:Строка r : Открытие файла для чтения. Файл должен существовать. *Доступен только режим r.</p>
Возвращаемое значение	Файловый объект браузера App Engine Содержимое файла можно читать с помощью функций readLine(), readAll().
Примечания	Функция openCommonFile() позволяет работать с файлами (чтение/запись) в общей области приложений, тогда как функция openFile() открывает для чтения файлы, расположенных в каталоге, в который установлено приложение.
Пример	<pre>var fileSystemObj = new FileSystem(); var fileObj = fileSystemObj.openFile('index.html', 'r'); alert(fileObj.readAll()); fileSystemObj.closeFile(fileObj);</pre>
Enumeration	None

## 9. Справка

### 9.1. Код страны

Код страны вы можете получить, используя строку `window.location.search`. См. раздел 6.7.

Страна	Код	Страна	Код
ALBANIA	AL	PORTUGAL	PT
ANDORRA	AD	ROMANIA	RO
AUSTRIA	AT	SERBIA	RS
BELGIUM	BE	SLOVAKIA	SK
BOSNIA	BA	SLOVENIA	SI
BULGARIA	BG	SPAIN	ES
CIS	RU	SWEDEN	SE
CROATIA	HR	SWITZERLAND	CH
CYPRUS	CY	UK	GB
CZECH	CZ	AMERICA	US
DENMARK	DK	CANADA	CA
FINLAND	FI	MEXICO	MX
FRANCE	FR	ARGENTINA	AR
GERMANY	DE	CHILE	CL
GREECE	GR	PERU	PE
HUNGARY	HU	THAILAND	TH
ICELAND	IS	VIETNAM	VN
IRELAND	IE	MALAYSIA	MY
ITALY	IT	INDONESIA	ID
LUXEMBOURG	LU	PHILIPPINES	PH
MACEDONIA	MK	BRAZIL	BR
MONTENEGRO	ME	ALGERIA	DZ
NETHERLAND	NL	MOROCCO	MA
NORWAY	NO	TUNISIA	TN
POLAND	PL		

### 9.2. Код языка

Код языка вы можете получить, используя строку `window.location.search` (см. раздел 6.7.). Приведенные ниже коды передаются в атрибут „lang”. В версии браузера App Engine 5.1 не поддерживаются языки Tamil и Hindi, для них используется код „en-GB”.

Язык	Код	Язык	Код
Afrikaans	af	Latvian	lv
Amharic	am	Lithuanian	lt
Arabic	ar	Lithuanian	lt
Bulgarian	bg	Norwegian	no
Chinese	zh-CN	Persian	fa
Croatian	hr	Polish	pl
Czech	cs	Portuguese	pt
Danish	da	Portuguese(America)	pt-US
Dutch	nl	Romany	ro
English	en	Russian	ru
English(GB)	en-GB	Serbian	sr

Estonian	et	Slovak	sk
Finnish	fi	Spanish	es
French	fr	Spanish(America)	es-US
French(America)	fr-US	Swahili	sw
German	de	Swedish	sv
Greek, Modern	el	Taiwan	zh-TW
Hausa	ha	Tamil (Not provided)	en-GB
Hebrew	he	Thai	th
Hindi (Not provided)	en-GB	Turkish	tr
Hongkong	zh-HK	Urdu	ur
Hungarian	hu	Vietnamese	vi
Igbo	ig	Xhosa	xh
Indonesian	id	Yoruba	yo
Italian	it	Zulu	zu
Korean	ko		

В следующей таблице приведены значения атрибута „language“, который в настоящее время заменен атрибутом „lang“. Т.к. атрибут „language“ в дальнейшем планируется удалить, рекомендуется его не использовать.

Язык	Код	Язык	Код
Korean	0	Spanish	24
English	1	Swedish	25
Spanish(America)	2	Turkish	26
French(America)	3	Chinese	27
Portuguese(America)	4	Hongkong	28
Bulgarian	5	Taiwan	29
Croatian	6	Japanese	30
Czech	7	Maori	31
Danish	8	CMN	32
Dutch	9	YUE	33
Finnish	10	Hindi	34
French	11	Estonian	35
German	12	Latvian	36
Modern Greek	13	Lithuanian	37
Hungarian	14	Arabic	38
Italian	15	Persian	39
Norwegian	16	QAA	40
English(GB)	17	AD	41
Polish	18	Catalan	42
Portuguese	19	VAL	43
Romany	20	Hebrew	44
Russian	21	OTHER	45
Serbian	22	Thai	46
Slovak	23		

### 9.3. Код региона

Код региона можете получить, используя строку window.location.search. См. раздел 6.7.

Код	Регион
KOR	Korea
USA	USA/Canada/Mexico

BRA	Brazil, Paraguay, Uruguay, Argentine
PANEURO	Europe
CHI	China
HKG	Hong Kong
ARB	Arab
PANNORDIG	Nordic
SOUTHEASTASIA	South-East Asia (Vietnam, Thailand, India, China, Iran, Israel, Central Asia, East Asia, Africa)
ASIA_ATV	South-East Asia (Vietnam, Thailand, India, China, Iran, Israel, Central Asia, East Asia, Africa)
ASIA_DTV	Australia , New Zealand, Singapore
TW	Taiwan, Columbia
NORTHAFRICA	TURKEY, MOROCCO, TUNISIA
EA_DTV	Indonesia, Malaysia, Republic of South Africa, Vietnam
CIS	CIS
PHI	Philippine
S_AFR_DTV	South Africa DTV

## 9.4. Категория продукта

Код продукта (устройства) вы можете получить, используя строку `window.location.search`. См. раздел 6.7.

Код	Продукт
0	TV
1	Monitor
2	Blue-ray Disk

## 9.5. Предупреждение

### 9.5.1. Чувствительное к регистру устройство

Broadcom Target является чувствительной к регистру системой. Поэтому необходимо обращать внимание на правильное написание имен. Например, если файл называется `player.js`, а в файле `index.html` вы напишете так, как показано ниже, вы получите ошибку в системе Broadcom.

```
<script language="javascript" type="text/javascript" src="Javascript/Player.js"></script>
```