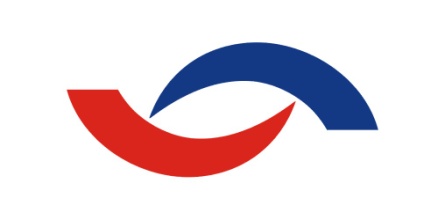
**密 级：内部公开**

**文档编号：GZSIMWARE\_PA\_REGU\_AZZN**

**版 本 号：V1.1**

**增强电视应用开发指南**

****

**广州西维尔计算机系统有限公司**

|  |  |
| --- | --- |
| 编制： | 生效日期： |
| 审核： | 批准： |

文档信息

|  |  |  |  |
| --- | --- | --- | --- |
| **文档名称** | 集团客户管理平台安装指南 | | |
| **电子文档** | 集团客户管理平台安装指南.doc/Microsoft WORD 2010 | | |
| **文件状态** | ■ **草稿** □ **正式发布** **正在修改** | | |
| **编 写 人** | 陈华春 | **日 期** | 2015年8月 31日 |
| **校 对 人** |  | **日 期** | 年月日 |
| **审 核 人** |  | **日 期** | 年月日 |
| **批 准 人** |  | **日 期** | 年月日 |

变更记录

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **变更序号** | **变更原因** | **变更页码** | **变更前版本号** | **变更后版本号** | **更改人** | **批准人** | **生效日期** | **备注** |
| **1** | 文档建立 |  |  | V1.0 | 陈华春 |  | 2015-8-31 |  |
| **2** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |
| **5** |  |  |  |  |  |  |  |  |
| **6** |  |  |  |  |  |  |  |  |

**目录**

[1. APP开发指南 7](#_Toc431306419)

[1.1. App开发规范 7](#_Toc431306420)

[1.2. package.json 7](#_Toc431306421)

[1.3. App开发步骤 8](#_Toc431306422)

[1.3.1. 新建package.json 8](#_Toc431306423)

[1.3.2. 配置调试选项 8](#_Toc431306424)

[1.3.3. 建html文件 10](#_Toc431306425)

[1.3.4. 新建css文件 10](#_Toc431306426)

[1.3.5. 新建js文件 10](#_Toc431306427)

[1.4. 接口说明 11](#_Toc431306428)

[1.4.1. 属性 11](#_Toc431306429)

[1.4.1.1. AppLoader.status 11](#_Toc431306430)

[1.4.1.2. AppLoader.app 11](#_Toc431306431)

[1.4.2. 方法 12](#_Toc431306432)

[1.4.2.1. AppLoader.offset([offset]) 12](#_Toc431306433)

[1.4.2.2. AppLoader. sendMessage(msg) 12](#_Toc431306434)

[1.4.2.3. AppLoader.install() 12](#_Toc431306435)

[1.4.2.4. AppLoader.show() 12](#_Toc431306436)

[1.4.2.5. AppLoader.hide() 12](#_Toc431306437)

[1.4.2.6. AppLoader.destroy() 12](#_Toc431306438)

[1.4.2.7. AppLoader.uninstall() 12](#_Toc431306439)

[1.4.2.8. AppLoader.active() 13](#_Toc431306440)

[1.4.2.9. AppLoader.unActive() 13](#_Toc431306441)

[1.4.2.10. AppLoader.bind(eventName, callback) 13](#_Toc431306442)

[1.4.2.11. AppLoader.unbind([eventId]) 13](#_Toc431306443)

[1.4.2.12. AppLoader.registerEvent(eventId) 13](#_Toc431306444)

[1.4.2.13. AppLoader.on(eventName, callback) 13](#_Toc431306445)

[1.4.2.14. AppLoader.off([eventName], [callback]) 14](#_Toc431306446)

[1.4.2.15. AppLoader.fire([eventName], [param]…) 14](#_Toc431306447)

[1.4.2.16. AppLoader.go(url, [param]) 14](#_Toc431306448)

[1.4.3. 事件 14](#_Toc431306449)

[1.4.3.1. status 14](#_Toc431306450)

[1.4.3.2. active 14](#_Toc431306451)

[1.4.3.3. unActive 14](#_Toc431306452)

[1.4.3.4. show 14](#_Toc431306453)

[1.4.3.5. hide 14](#_Toc431306454)

[1.4.3.6. destroy 15](#_Toc431306455)

[2. APP引擎集成指南 15](#_Toc431306456)

[2.1. 集成步骤 15](#_Toc431306457)

[2.1.1. 引入App加载引擎 15](#_Toc431306458)

[2.1.2. 配置引擎（可选） 15](#_Toc431306459)

[2.1.3. 初始化 16](#_Toc431306460)

[2.1.4. 更新终端状态 16](#_Toc431306461)

[1.5. 接口说明 16](#_Toc431306462)

[1.5.1. ETV.config(options) 16](#_Toc431306463)

[1.5.2. ETV.Engine 17](#_Toc431306464)

[1.5.2.1. ETV.Engine.init(options) 17](#_Toc431306465)

[1.5.3. 方法 17](#_Toc431306466)

[1.5.3.1. ETV.Engine.sendStatus() 17](#_Toc431306467)

[1.5.3.2. ETV.Engine.sendMessage(message) 18](#_Toc431306468)

[1.5.3.3. ETV.Engine.setStatus(status) 18](#_Toc431306469)

[1.5.3.4. ETV.Engine.getStatus() 19](#_Toc431306470)

[1.5.3.5. ETV.Engine.setPlayState(state) 19](#_Toc431306471)

[1.5.3.6. ETV.Engine.setChannelId(id) 19](#_Toc431306472)

[1.5.3.7. ETV.Engine.setAssetId(id) 19](#_Toc431306473)

[1.5.3.8. ETV.Engine.setPoint(point) 19](#_Toc431306474)

[1.5.3.9. ETV.Engine.getLoader(app) 19](#_Toc431306475)

[1.5.3.10. ETV.Engine.findLoader(appId) 20](#_Toc431306476)

[1.5.3.11. ETV.Engine.getLoaders() 20](#_Toc431306477)

[1.5.3.12. ETV.Engine.on(eventName, callback) 20](#_Toc431306478)

[1.5.3.13. ETV.Engine.off([eventName], [calllback]) 20](#_Toc431306479)

[1.5.3.14. ETV.Engine.fire(eventName, [param]…) 20](#_Toc431306480)

[1.5.4. 事件 21](#_Toc431306481)

[1.5.4.1. ready 21](#_Toc431306482)

[1.5.4.2. install 21](#_Toc431306483)

[1.5.4.3. show 21](#_Toc431306484)

[1.5.4.4. hide 21](#_Toc431306485)

[1.5.4.5. destroy 21](#_Toc431306486)

[1.5.4.6. uninstall 21](#_Toc431306487)

[1.5.4.7. change 21](#_Toc431306488)

[1.5.4.8. appInstall 22](#_Toc431306489)

[1.5.4.9. appShow 22](#_Toc431306490)

[1.5.4.10. appHide 22](#_Toc431306491)

[1.5.4.11. appDestroy 22](#_Toc431306492)

[1.5.4.12. appUninstall 22](#_Toc431306493)

[1.5.4.13. appInit 22](#_Toc431306494)

[1.5.4.14. appActive 22](#_Toc431306495)

[1.5.4.15. appUnActive 23](#_Toc431306496)

[1.5.4.16. appAllUnActive 23](#_Toc431306497)

[1.5.4.17. socketConnect 23](#_Toc431306498)

[2.2. 常见问题 23](#_Toc431306499)

[2.2.1. App 调用不到中间件接口 23](#_Toc431306500)

[2.2.2. App用到的按键和页面冲突 23](#_Toc431306501)

# APP开发指南

## App开发规范

App是一个单页面应用，开发建议遵从结构、表现、行为分离。Javascript 编码必须遵从CMD 规范。 App引擎采用SeaJs 作模块化加载，关于SeaJs和CMD规范请参考<http://seajs.org/docs/>

## package.json

app包根目录必须包含app的配置文件 package.json，必须包含的节点说明

|  |  |  |
| --- | --- | --- |
| 元素 | 说明 | 类型 |
| name | 应用名称 | 字符串 |
| version | 版本号 | 字符串 |
| author | 应用的作者 | 字符串 |
| main | 应用运行入口配置，包含3个节点：css、html、js | 对象 |
| css | app的css文件相对地址 | 字符串或数组 |
| html | app的html模版相对地址 | 字符串 |
| js | app的js文件相对地址 | 字符串或数组 |

**示例：**

|  |
| --- |
| {  "name": "测试应用",  "version": "0.0.1",  "author": "西维尔",  "description": "测试使用",  "main": {  "css": "res/style.css",  "html": "res/index.html",  "js": "res/app.js"  }  } |

## App开发步骤

### 新建package.json

按照App开发规范新建package.json文件

### 配置调试选项

为了方便调试，可以在package.json中增加测试节点，可以在终端模拟器上运行app预览，例如：

|  |
| --- |
| {  "name": "测试应用",  "version": "0.0.1",  "author": "西维尔",  "description": "测试使用",  "main": {  "css": "res/style.css",  "html": "res/index.html",  "js": "res/app.js"  },  "test": {  "status": {  "areaCode": "GZ",  "pageId": "",  "channelId": "",  "columnId": "",  "assetId": ""  },  "init": {  "auto": true,  "width": 1050,  "height": 700,  "left": 0,  "top": 0,  "active": true,  "startPoint": [],  "stopPoint": [],  "startTime": [],  "stopTime": [],  "playState" :[]  }  }  } |

status 是用来模拟投放策略的触发条件，如果终端的状态和status匹配就按照init节点描述的启动选项启动app。

**status节点说明：**

|  |  |  |
| --- | --- | --- |
| 元素 | 说明 | 类型 |
| areaCode | 区域码 | 整型或字符串 |
| pageId | 页面标识 | 字符串 |
| channelId | 频道Id | 整型或字符串 |
| columnId | 栏目标识 | 字符串 |
| assetId | 媒资标识 | 字符串 |

**init 节点说明：**

|  |  |  |
| --- | --- | --- |
| 参数名称 | 描述 | 类型 |
| width | APP初始展示宽度 | 整型 |
| height | APP初始展示高度 | 整型 |
| left | APP初始展示x坐标 | 整型 |
| top | APP初始展示y坐标 | 整型 |
| auto | 是否自动展示，如果是，按 startTime/ startPoint开始展示，如果startTime/ startPoint 没设置，安装完立即展示，  如果false，在在后台运行。 | 布尔值 |
| startTime | 开始展示时间，auto 为true才有效，可数组  格式： [“yyyy-MM-dd hh:mm:ss”] | 数组 |
| stopTime | 展现结束时间，auto 为true才有效，可数组  格式：[“yyyy-MM-dd hh:mm:ss”] | 数组 |
| active | 展现时是否处于活动状态，为true时才能接收按键事件 | 布尔值 |
| activeKey | 激活和取消激活状态的按键值 | 整型 |
| showKey | 显示和隐藏按键值 | 整型 |
| playState | 播放器状态（0 等待，1播放中，2暂停，3停止，4快进，5快退，6音量调节），可数组  格式：[2, 3] | 数组 |
| startPoint | 展示播放器时间点，可数组，相对时间  格式：[2, 30] | 数组 |
| stopPoint | 停止播放器时间点，可数组，相对时间  格式：[10, 60] | 数组 |

### 建html文件

新建html文件，表现app的结构html代码

### 新建css文件

新建css文件，表现app外观css代码

### 新建js文件

新建js文件，按CMD规范编写代码，代码样例：

|  |
| --- |
| define(function (require, exports, module) {  var app = {  init: function () {  //do something  },  destroy: function () {  //do something  }  };  module.exprots = app;  }); |

Js模块需要实现2个接口，init 和 destroy。不需要自己手动调用，app加载引擎会自动调用。

init 是程序入口，初始化函数

destroy是app销毁函数，在app销毁时有app加载引擎自动运行，如果app需要在app销毁时做资源的释放，可以在这个函数里面实现。

App在启动时，app加载引擎会自动注入TVUI和AppLoader，因此可以在js直接调用TVUI和AppLoader， 不需要自己手动加载。

注意，由于app加载引擎采用动态创建的iframe的方式来实现无阻塞加载，js是不完全支持中间件的接口，仅支持由TVUI提供的中间件接口。

TVUI相关接口请参考 TVUI使用手册

## 接口说明

该接口是提供给APP内部调用，在app启动时，引擎会自动注入AppLoader对象，app的js可以直接调AppLoader，不需要实例化

AppLoader继承TVUI.Base， 拥有TVUI.Base的全部属性和方法，详细请参考TVUI使用手册

App内部无需引用TVUI，引擎自动注入TVUI。可以直接调用

### 属性

#### AppLoader.status

终端环境状态对象

#### AppLoader.app

装载的app信息对象

### 方法

#### AppLoader.offset([offset])

设置/获取iframe的位置、大小 ,offset格式：

｛width：100， height：100， left：100， top: 100｝

#### AppLoader. sendMessage(msg)

向服务器发送消息，msg说明

消息对象，必须包含2个属性，type 和 msg，如：

{type: 1, msg: '安装成功'}

#### AppLoader.install()

安装app， 不需要手动运行，引擎会自动运行

#### AppLoader.show()

显示app

#### AppLoader.hide()

隐藏app

#### AppLoader.destroy()

销毁app

#### AppLoader.uninstall()

卸载app

#### AppLoader.active()

激活app， 激活后可以接收遥控器事件

#### AppLoader.unActive()

取消激活app，取消激活后，不能接收遥控器事件

#### AppLoader.bind(eventName, callback)

侦听引擎事件。和使用 ETV.Engine.on(eventName, callback) 效果一样，但用 bind的方式在app销毁时会自动销毁事件， 用on即要手动销毁。

#### AppLoader.unbind([eventId])

销毁绑定引擎的事件，evnetId可选， 如不传，即销毁全部事件

#### AppLoader.registerEvent(eventId)

注册dom事件，如按键事件， 在app销毁时自动销毁，不需要手工销毁，用法如下：

|  |
| --- |
| AppLoader.registerEvent(TVUI.Event.onKey(TVUI.Key.NUM1, function(e){  //do something  })); |

当app销毁后，这个事件会自动销毁，如果不用这个方法注册事件，就需要手动去销毁，否则会导致内存泄漏

#### AppLoader.on(eventName, callback)

绑定事件

#### AppLoader.off([eventName], [callback])

销毁事件

#### AppLoader.fire([eventName], [param]…)

触发事件

#### AppLoader.go(url, [param])

页面调转

#### AppLoader.addKey(code)

注册按键，参数code是按键的键值，可以是数字类型或数组。

#### AppLoader.removeKey(code)

删除按键，参数code是按键的键值，可以是数字类型或数组。

#### AppLoader.getExt(name)

获取引擎扩展属性

### 事件

#### status

终端状态发生变化时触发

#### active

app激活时触发

#### unActive

app取消激活时触发

#### show

app显示时触发

#### hide

app隐藏时触发

#### destroy

app销毁时触发

# APP引擎集成指南

## 集成步骤

### 引入App加载引擎

采用标签引入app引擎 文件名 etv.engine.min.js 。该文件包含了TVUI、和 socket.io

### 配置引擎（可选）

加载完引擎后，按需要配置选项

配置方法如：

|  |
| --- |
| ETV.config({  loader: 'js/loader.min.js',  seajs: 'js/sea.js',  socket: 'http://172.16.1.10:8011/',  apps: 'http://172.16.1.10/APPS/'  }); |

配置都有默认值，需要修改配置才运行修改

**说明**

|  |  |  |
| --- | --- | --- |
| 配置项 | 说明 | 默认值 |
| loader | Loader模块js地址，该文件的作用是无阻塞加载app | http://172.16.1.10/APPS/engine/js/loader.min.js |
| seajs | Seajs文件地址，用作模块化加载app | http://172.16.1.10/APPS/engine/js/sea.js |
| socket | WebSocket服务器地址 | http://172.16.1.10:8011/ |
| apps | Apps服务器地址 | http://172.16.1.10/APPS/ |

### 初始化

执行初始化, 例如:

|  |
| --- |
| ETV.Engine.init({  areaCode: 101,  pageId: "",  channelId: "",  columnId: "",  assetId: ""  }); |

参数全部是可选的，按需设置。可不传任何参数如： ETV.Engine.init()

areaCode自动读取机顶盒

**说明**

|  |  |  |
| --- | --- | --- |
| 参数 | 说明 | 类型 |
| areaCode | 区域码 | 整型或字符串 |
| pageId | 页面标识 | 字符串 |
| channelId | 频道Id | 整型或字符串 |
| columnId | 栏目标识 | 字符串 |
| assetId | 媒资标识 | 字符串 |

### 更新终端状态

如终端状态发生了变化，需要使用app引擎提供的接口，通知app引擎。例如：更换频道、更换栏目、播放器状态变化，播放时间变化等。

如更换切换频道，调用 ETV.Engine.setChannelId(1001);

详细接口请参考接口说明

## 接口说明

### ETV.config(options)

APP引擎配置方法，如需要修改配置，在ETV.Engine.init() 之前执行修改配置， options参数如下：

|  |  |  |
| --- | --- | --- |
| 参数名称 | 描述 | 使用情况 |
| socket | Websocket 服务器地址，如：http://172.16.1.10:8011/ | ？ |
| apps | Apps服务器地址 | ？ |
| loader | App加载器地址 | ？ |
| icNo | 缺省的icNo | ？ |
| areaCode | 缺省的区域码 | ？ |

### ETV.Engine

引用方式，在portal页面应用 etv.engine.min.js 即可，该文件包含了TVUI

#### ETV.Engine.init(options)

引擎初始化

options说明，所有参数都是可选的，在引用etv.engine.js 后需要运行初始化

|  |  |  |
| --- | --- | --- |
| 参数名称 | 描述 | 使用情况 |
| pageId | 页面标识 | １ |
| channelId | 频道标识 | ？ |
| columnId | 栏目标识 或 影片类型标识 | ？ |
| assetId | 媒资Id 或 节目单id | ？ |

### 方法

#### ETV.Engine.sendStatus()

向服务器发送状态，当终端状态发生改变的时候，引擎会自动发送状态， 通常不需要自己手动运行，除非你想手动。

#### ETV.Engine.sendMessage(message)

向服务器发送消息，message参数如下：

|  |  |  |
| --- | --- | --- |
| 参数名称 | 描述 | 使用情况 |
| type | 消息类型（1安装成功、2显示成功、3隐藏成功、4销毁成功、5卸载成功），1对应后端的install，5对应后端的uninstall | 1 |
| msg | 消息内容 | 1 |

当app的状态发送变化，引擎会自动发送，通常不需要自己手动运行

#### ETV.Engine.setStatus(status)

status说明

|  |  |  |
| --- | --- | --- |
| 参数名称 | 描述 | 使用情况 |
| icNo | 智能卡号 | ？ |
| areaCode | 区域码 | ？ |
| pageId | 页面标识 | ？ |
| channelId | 频道标识 | ？ |
| columnId | 栏目标识 或 影片类型标识 | ？ |
| assetId | 媒资Id 或 节目单id | ？ |
| point | 媒资播放当前时间点 | ？ |
| playState | 播放器状态（待播放，播放中，暂停，停止，快进，快退，音量） | ？ |

设置状态后，会自动运行ETV.Engine.sendStatus()

#### ETV.Engine.getStatus()

获取终端状态，返回状态对象

#### ETV.Engine.setPlayState(state)

设置播放器状态

0 播放器等待播放

1 播放中

2 暂停

3 停止

4 快进

5 快退

6 音量调节

#### ETV.Engine.setChannelId(id)

设置状态的频道id

#### ETV.Engine.setAssetId(id)

设置状态的媒资id

#### ETV.Engine.setPoint(point)

设置状态媒资的播放时间点

#### ETV.Engine.getLoader(app)

从缓存中获取AppLoader，如果缓存不存在app的加载器，即实例化创建。

参数app，即webSocket指令中install的参数

#### ETV.Engine.findLoader(appId)

根据appId从缓存中获取AppLoader，如果不存在，返回undefined

当portal需要控制app的时候，先通过这个方法获取app对应的AppLoader，再根据AppLoader提供的接口操作app

#### ETV.Engine.getLoaders()

从缓存中获取全部的AppLoader

#### ETV.Engine.on(eventName, callback)

绑定事件，如当给app显示成功绑定事件，代码如下：

|  |
| --- |
| ETV.Engine.on(‘appShow’, function(AppLoader){  //当app显示的时候，就会运行这里的代码，并且得到是哪个AppLoader触发的  }); |

#### ETV.Engine.off([eventName], [calllback])

销毁事件，有三种用法：

1. ETV.Engine.off(eventName, callback) 删除具体的一个事件
2. ETV.Engine.off(eventName) 删除具体一类事件
3. ETV.Engine.off() 删除引擎全部事件，不推荐调用

#### ETV.Engine.fire(eventName, [param]…)

触发事件，如：

ETV.Engine.fire(‘eventName1’) 触发事件名称为 eventName1的事件

ETV.Engine.fire(‘eventName1’, ‘abc’) 触发事件名称为 eventName1的事件,并且传递一个参数 abc

#### ETV.Engine.setExt(name, value)

设置扩展属性

### 事件

#### ready

当app引擎完成初始化时触发

#### install

收到安装指令时触发，参数app对象

#### show

收到show指令时触发，参数app对象

#### hide

收到hide指令时触发， 参数appId

#### destroy

收到destroy时触发，参数appId

#### uninstall

收到uninstall指令时触发，参数appId

#### change

当状态发生变化时触发，参数 status对象

#### appInstall

app安装完成时触发，参数AppLoader

#### appShow

app显示完成时触发，参数AppLoader

#### appHide

app隐藏完成时触发，参数AppLoader

#### appDestroy

app销毁完成时触发，参数AppLoader

#### appUninstall

app卸载完成时触发，参数AppLoader

#### appInit

app初始化完成时触发，参数AppLoader

#### appActive

app激活时触发，参数AppLoader

#### appUnActive

app取消活动时触发，参数AppLoader

#### appAllUnActive

全部的app都取消活动时触发

#### socketConnect

WebSocket建立时触发

## 常见问题

### App 调用不到中间件接口

由于app是动态的无阻塞加载，app是不能直接调用中间件接口的，如需要调中间件接口需要通过TVUI.API提供的中间件接口调用。

TVUI.API 包含 DataAccess、SysSetting、CA、Network、FileSystem、user、EPG 可满足常用功能开发需求

### App用到的按键和页面冲突

例如app用到返回键，页面也用到了返回键，app在活动状态时，按返回键就时app和页面都会响应。

**解决方法：**

1）、如果页面是使用TVUI开发。增加以下事件处理

|  |
| --- |
| //当app销毁或取消活动事，激活页面，恢复页面接收事件  ETV.Engine.on('appUnActive appDestroy', function(){  \_\_page.active();  });  //当app被激活时，禁止页面接收事件  ETV.Engine.on('appActive', function(){  \_\_page.unActive();  }); |

2）、如果页面不是使用TVUI的，可以用变量来记录app状态

|  |
| --- |
| var appActive = true;  ETV.Engine.on('appActive', function(){  appActive = true;  });  ETV.Engine.on('appUnActive appDestroy', function(){  appActive = false;  });  //页面程序根据 appActive 来做判断。判断是否响应按键事件 |

3）、如果页面是使用 document.onkeypress 的方式绑定事件，如果app用到的事件与页面的冲突，在app中使用按键事件前先注册，不在需要这个按键时删除。

注册按键方法：AppLoader.addKey(code)

删除按键方法：AppLoader.removeKey(code)