РАЗРАБОТКА ПЛАГИНОВ ДЛЯ ПРИЛОЖЕНИЯ «МЕДИАЦЕНТР» (Android)

Версия 1.0.5 (версия приложения 0.2.4)

1. ОБЩИЕ СВЕДЕНИЯ

1.1 ОПИСАНИЕ СТРУКТУРЫ ПЛАГИНА

Плагин представляет собой **ZIP архив** (упакованный стандартным способом, режим сжатия – **deflate**) с расширением *.IMC.zip, который содержит следующие файлы:

service.xml (обязателен) – конфигурационный файл плагина, содержит шаблон настроек плагина, используемые строковые переменные на разных языках и т.д.

service.lua (обязателен) – основной модуль программы, написанный на языке Lua

icon.png (необязателен) – иконка плагина (отображается на экране)

Также архив может содержать дополнительные модули и картинки.

1.2 ОПИСАНИЕ КОНФИГУРАЦИОННОГО ФАЙЛА

Конфигурационный файл service.xml представляется собой XML файл следующего вида:

```
<?xml version="1.0" encoding="UTF-8"?>
<service>
<id>[Идентификатор плагина, строка без пробелов]</id>
<preferences>
[Перечисление настроек плагина]

</preferences>

<strings>
[Строки/массивы строк на базовом (английском) языке]
</strings>

<strings-[идентификатор локализации (ru,de,ua)]>
[Строки/массивы строк на языке локализации]
</strings-ru>

</service>
```

Настройки плагина могут быть следующими:

```
<item name="username" label="@string/username" type="text" />
<item name="password" label="@string/password" type="password"  />
```

```
<item name="list" label="@string/list" type="list" entries="@array/list_ids" values="@array/list_vls"  />
```

Параметры: name – имя настройки, label–заголовок на экране, type– тип элемента (текст, пароль, выбор из списка), entries–строки на экране, values–возвращаемые значения.

Строки и массивы строк:

```
<string name="label">Kartina.TV</string>
<string-array name="qualities">
        <item>Eco</item>
        <item>SD</item>
        <item>HD</item>
</string-array>
```

Параметры: name – имя строки/массива строк. Доступ к строке осуществляется через @string/[имя строки]. Доступ к массиву строк осуществляется через @array/[имя массива строк].

1.3 ОПИСАНИЕ ОСНОВНОГО МОДУЛЯ

Основной модуль service.lua представляет собой текстовый файл, сохраненный в кодировке UTF-8 (без BOM), содержащий программный код на языке Lua (версия 5.2.1). При работе модуля доступны стандартные функции языка Lua. Подробнее про язык Lua можно прочитать на следующих ресурсах:

http://www.lua.org/manual/5.2/

http://www.lua.ru/doc/

Рассмотрим функции, используемые приложением.

При загрузке плагина вызывается функция «onLoad», данная функция должна вернуть номер версии среды исполнения (на данный момент цифра 1). Если номер версии не поддерживается средой исполнения, то плагин будет выгружен.

```
function onLoad()
   log.i("Hello from example!")
   return 1
end
```

При завершении работы плагина вызывается функция «onUnLoad»:

```
function onUnLoad()
   log.i("Bye bye!!!")
```

```
    end
```

При формировании списка контента/запросе ссылки для воспроизведения вызывается функция «onCreate». В качестве параметра передается таблица со значениями параметров, переданных в адресе mrl. Данная функция должна вернуть таблицу со значениями элементов списка.

```
function onCreate(context)
    local t={view="grid",type="folder"}
      if context.id == "1" then
        table.insert(t, {title="Hello from one element!", mrl="#folder/"})
      end

table.insert(t,{title="@string/one",mrl="#folder/id=1",image="#self/icon.png"

table.insert(t,{title="@string/two",mrl="#folder/id=2",image="#self/icon.png"

table.insert(t,{title="@string/three",mrl="#folder/id=3",image="#self/icon.png"
      return t
    end
```

Данный пример выводит список из 3 элементов в виде иконок с текстом.

Рассмотрим основные параметры таблицы списка воспроизведения:

view	ВИДЫ ОТОБРАЖЕНИЯ СПИСКА ЭЛЕМЕНТОВ: simple – простой список элементов list – список (иконка, заголовок и описание), grid–небольшие иконки с заголовком, grid_large – большие иконки с заголовками, grid_poster – иконки в виде обложек фильмов, annotation – страница с описанием контента ВИДЫ ОТОБРАЖЕНИЯ ВСПЛЫВАЮЩЕЙ ИНФОРМАЦИИ: keyword – ввести ключевое слово (значение по умолчанию передается в параметре keyword) login – ввод пользователя и пароля (значения по умолчанию передается в параметрах username,password, в параметре message задается подсказка, в параметре error информация об ошибке) select – выбор значения из передаваемого списка (значения mrl) msgbox – сообщение пользователю error – сообщение об ошибке setup – открыть настройки плагина refresh – обновить текущий список playback–запуск управляемого воспроизведения видео (параметры: label–заголовок контента, mrl–URL адрес видео

	потока, previous – mrl ссылка для получения предыдущего видео, next–mrl ссылка для получения следующего видео, meta – mrl ссылка вызова информирования плагина о процессе воспроизведения, seekable–если передана строка "true" то будет разрешено позиционирование видео, direct–если передана строка "true" то URL адрес будет считаться прямой ссылкой на контент)
type	Тип контента: folder – каталог (рекомендуется указывать когда в списке элементов может быть только один элемент) playlist – плейлист stream – онлайн трансляция video – видеофайл audio – аудио файл/интернет-радио
keyword	ключевое слово по умолчанию
username	пользователь по умолчанию
password	пароль по умолчанию
message	текст подсказки
error	текст ошибки при отображении всплывающей информации

Рассмотрим основные поля значений таблицы списка:

title	заголовок элемента списка
mrl	ссылка на контент, формируется следующим образом: #тип/[имя1=значение&имя2=значение2&…] где тип может быть: folder–каталог playlist – плейлист stream – ссылка на онлайн ТВ video–ссылка на видео файл audio – ссылка на аудио файл/радио аргументы имя1,имя2,…,имяN со значениями передаются в функцию onCreate как массив
image	ссылка на иконку, для ссылки на картинку внутри архива плагина необходимо использовать следующий формат: #self/[имя файла в архиве]
annotation	описание контента
group	имя группы
meta	форматная строка, описывающая получение дополнительной информации о контенте (телепрограмма и т.д.), принцип построения: #[jtv|self]/[аргументы] , где: jtv – использование встроенного механизма заполнения телепрограммы в формате JTV, аргументы: url–ссылка на архив

	JTV в ZIP формате, name–имя канала в архиве, shift–часовой пояс телепрограммы self – использование механизма реализованного в плагине, аргументы на усмотрении разработчика плагина (кроме имени type) при запросе информации к аргументам прибавляется аргумент «type» со значением типа запрашиваемой информации: annotation – получение информации об элементе списка EPG – получение списка телепрограммы канала
size	размер файла/носителя
free	свободно

При запросе информации вызывается функция «onMeta». В качестве параметра передается таблица со значениями аргументов, переданных в параметре «meta».

```
function onMeta (context)
   assert(context.type == "EPG")
   local t={ }
   table.insert(t,{title="News 24",time1=1234324})
   return t
end
```

Рассмотрим основные свойства таблицы «annotation»:

title	заголовок элемента списка
image	ссылка на иконку
group	имя группы
annotation	описание элемента списка
size	размер файла/носителя
free	свободно

Обязательно для заполнения только одно поле из перечисленных выше.

Рассмотрим основные колонки таблицы «EPG»:

title	заголовок телепрограммы
time1	время начала в формате unix timestamp
time2	(необязателен) время завершения в формате unix timestamp

При управляемом воспроизведении видео (view = playback) вызывается функция «onEvent». В качестве параметра передается таблица со значениями аргументов, переданных в параметре «meta».Данная функция может использоваться для отправки статистики.

```
function onEvent(e)
   if e.action == "got" then
      log.i('готов к воспроизведению')
   elseife.action == "update" then
      log.i('Время '..tostring(e.time)..'сек. '..tostring(e.percent)..' %')
   end
end
```

Рассмотрим системные свойства таблицы аргументов (e):

action	событие: got – видео готово к воспроизведению (вызывается перед воспроизведением) watched – видео просмотрено update – обновление информации о воспроизведении (каждые 1000мс)
time	время воспроизведения в секундах
percent	процент воспроизведения

2. РАСШИРЕНИЕ ЯЗЫКА LUA

2.1 ОСНОВНЫЕ ФУНКЦИИ

plugin.probe(arg1[, arg2 [, arg3[,…]]])

Загрузка дополнительных модулей, если модуль не найден в архиве плагина то загружается общий модуль.

Возвращаемое значение: булево

Пример (загрузка общего модуля video):

```
plugin.probe("video")
```

plugin.engine(arg1[, arg2 [, arg3[,…]]])

Получение параметров среды исполнения (version – версия приложения, locale–локализация, tmp–каталог временных файлов, language – язык, country – страна, model – модель устройства, vod – есть ли поддержка M3U VOD).

Возвращаемое значение: строка, количество равно количеству переданных аргументов

Пример:

```
local v=plugin.engine("version")
```

plugin.node(arg1[, arg2 [, arg3[,…]]])

Получение параметров плагина из конфигурационного файла service.xml(идентификатор, описание и т.п.).

Возвращаемое значение: строка, количество равно количеству переданных аргументов

Пример:

```
local id=plugin.node("id")
```

plugin.decode(arg1[, arg2 [, arg3[,…]]])

Декодирование строковых представлений данных плагина (получение строк и массивов строк по идентификаторам).

Возвращаемое значение: строка или таблица, количество равно количеству переданных аргументов

Пример:

```
local s,arr=plugin.decode("@string/username","@array/defaults")
```

plugin.prop(name1[,name2 [,name3 [,…]]])

Получение значений настроек плагина.

Возвращаемое значение: строка, количество равно количеству переданных имен параметров

Пример:

```
local user,pass=plugin.prop("username","password")
```

plugin.setprop(name1, value1[,name2, value2 [,name3, value3 [,…]]])

Установка значений настроек плагина.

Пример:

```
plugin.setprop("username","123","password", "321")
```

plugin.sysprop(name1[,name2 [,name3 [,…]]])

Получение значений настроек (приложения и других сервисов).

Возвращаемое значение: строка, количество равно количеству переданных имен параметров

Пример:

```
local region,server=plugin.sysprop("youtube_region","udpxy_server")
```

2.2 ФУНКЦИИ ОТЛАДКИ

log.v(text) или print(text)

Добавление в журнал регистрации отладочной информации.

Пример:

```
log.v("Hello from me!")
```

log.i(text)

Добавление в журнал регистрации информации о процессе работы плагина.

Пример:

```
log.i("Hello from me!")
```

log.w(text)

Добавление в журнал регистрации информации требующей внимания.

Пример:

```
log.w("Can’t parsing html page!")
```

log.e(text)

Добавление в журнал регистрации информации об ошибках.

Пример:

```
log.e("Error parsing html page!")
```

2.3 РАБОТА С ТЕКСТОВОЙ ИНФОРМАЦИЕЙ

string.ulen(text)

Подсчет количества символов в UTF-8 строке

Возвращаемое значение: число

Пример:

```
local len=string.ulen("Привет")
```

string.usub(text, start[, length])

Получить подстроку из UTF-8 строки

Возвращаемое значение: строка

Пример:

```
local sub=string.usub("Привет",2,3)
```

string.ucmp(text1, text2)

Сравнение UTF-8 строк.

Возвращаемое значение: число, если text1 <text2 тогда -1, если равны то 0 иначе 1

Пример:

```
local compare=string.ucmp("Привет","Рыбка")
```

string.ucasecmp(text1, text2)

Сравнение UTF-8 строк без учета регистра букв латинского и русского алфавита.

Возвращаемое значение: число, если text1 <text2 тогда -1, если равны то 0 иначе 1

Пример:

```
local compare=string.ucasecmp("Привет","привет")
```

string.ulower (text)

Преобразование UTF-8 строки в нижний регистр.

Возвращаемое значение: строка

Пример:

```
local hello=string.ulower("ПРИВЕТ")
```

string.uupper (text)

Преобразование UTF-8 строки в верхний регистр.

Возвращаемое значение: строка

Пример:

```
local hello=string.uupper("привет")
```

string.ucheck (text)

Проверка строки на кодировку UTF-8.

Возвращаемое значение: булево

Пример:

```
local isutf8=string.ucheck("привет")
```

2.4 КОДИРОВАНИЯ/ДЕКОДИРОВАНИЯ ИНФОРМАЦИИ

base64.decode(text)

Декодирование закодированной строки по методу BASE64

Возвращаемое значение: строка

Пример:

```
local decoded=base64.decode("gHJGHJds7898=")
```

base64.encode(text)

Кодирование строки по методу BASE64

Возвращаемое значение: строка

Пример:

```
local encoded=base64.encode("username:password")
```

md5.sum(text)

Получение хеш строки по MD5 алгоритму.

Возвращаемое значение: строка

Пример:

```
local hash=md5.sum("login/password")
```

md5.init()

Создание объекта формирования MD5 хеш строки.

Возвращаемое значение: объект

Методы объекта:

append(text) – добавление строки

finish() – получение хеш строки

Пример:

```
local t=md5.init()
t:append('Hello ')
t:append('World!')
local hash=t:finish()
```

sha1.sum(text)

Получение хеш строки по SHA1 алгоритму.

Возвращаемое значение: строка

Пример:

```
local hash=sha1.sum("login/password")
```

sha1.init()

Создание объекта формирования SHA1 хеш строки.

Возвращаемое значение: объект

Методы объекта:

append(text) – добавление строки

finish() – получение хеш строки

Пример:

```
local t=sha1.init()
t:append('Hello ')
t:append('World!')
local hash=t:finish()
```

2.5 СЕТЕВЫЕ ФУНКЦИИ

socket.tcp()

Создание TCP сокета

Возвращаемое значение: объект сокета

Пример:

```
local s=socket.tcp()
```

Методы сокет объекта:

connect(hostname[, port[, timeout]])

Соединение с сервером hostname.

Возвращаемое значение: булево

Пример:

```
local connected=s:connect("yandex.ru")
```

read([length])

Чтение текстовых данных из сокета длиной length (по умолчанию 4096 байт)

Возвращаемое значение: строка

Пример:

```
local text=s:read(200)
```

write(text)

Отправка текстовых данных.

Пример:

```

```

```
s:write("HELLO")
```

connected()

Проверка установлено ли соединение с сервером

Возвращаемое значение: булево

Пример:

```
local c=s:connected()
```

Функции передачи данных по HTTP протоколу:

http.get(url[,headers[,response]])

Загрузка содержимого интернет страницы по методу GET, параметры: headers– таблица дополнительных заголовков для передачи серверу, response– переменная (таблица) для записи ответа сервера (код и заголовки, код доступен через ключ http_code)

Возвращаемое значение: строка

Пример:

```
local resp={}
local t=http.get("http://yandex.ru/q=abracadabra",
{Referer="http://yandex.ru/"}, resp)
for k,v in pairs(resp) do
log.i("header "..tostring(k).."  "..tostring(v))
end
```

http.post(url[,headers[,data[,response]]])

Аналогично функции http.get, но формирует запрос к серверу методом POST. Параметр data – данные передаваемые после HTTP заголовка

Возвращаемое значение: строка

Пример:

```
local resp={}
```

```
	local t=http.post("http://yandex.ru/", {Referer="http://yandex.ru/"},
"q=abracadabra", resp)
	for k,v in pairs(resp) do
	log.i("header "..tostring(k).."  "..tostring(v))
	end
```

http.decode(query)

Разбор строки параметров.

Возвращаемое значение: таблица

Пример:

```
	local p=http.decode("name=user&value=kukareku")
```

http.urldecode(query)

Декодирование URL строки

Возвращаемое значение: строка

Пример:

```
	local d=http.urldecode("ku+ka+re+ku")
```

http.urlencode(query[,keep])

Кодирование URL строки

Возвращаемое значение: строка

Пример:

```
	local e=http.urlencode("ku ka re ku")
```

http.urlparse(url)

Разбор URL адреса

Возвращаемое значение: таблица, где:

scheme – протокол, folder–каталог, username - пользователь, password - пароль, hostname – имя сервера, port - порт, filepath - путь, query – строка параметров, fragment - параграф

Пример:

```
local u=http.urlparse ("http://youtube.com/watch?v=789hjkhuy8")
log.i("scheme "..u.scheme)
```

2.6 РАБОТА С ДОКУМЕНТАМИ В JSON ФОРМАТЕ

json.decode(text)

Преобразование JSON текста в таблицу

Возвращаемое значение: таблица

Пример:

```
local a="[1,2,3,4,5]"
local t=json.decode(a)
for k,v in pairs(t) do
log.i("item "..tostring(k).."  "..tostring(v))
end
```

2.7 РАБОТА С ДОКУМЕНТАМИ В XML ФОРМАТЕ

xml.load(url[,charset])

Загрузка XML документа из URL источника. Параметр charset – строка название кодировки документа (по умолчанию распознаются кодировки UTF-8,WINDOWS-1251,KOI8-R). Если загружаемый документ в кодировке UTF-8, то рекомендуется указать это явно в параметре charset.

Возвращаемое значение: xml объект

Пример:

```
local x=xml.load("http://gdata.youtube.com/feeds/api/standardfeeds/top_rated")
```

xml.parse(text)

Загрузка XML документа из текстовой строки.

Возвращаемое значение: xml объект

Пример:

```
local text=http.get("http://gdata.youtube.com/feeds/api/standardfeeds/top_rated")
local x=xml.parse(text)
```

Методы объекта xml:

count (name1[[,index1][,name2[,index2]][,…],nameN)

Получение количество элементов nameN в заданной иерархии. Если index не задан, то берется первый элемент.

Возвращаемое значение: число

Пример:

```
local x=xml.load("playlist.xspf")
local k=x:count("playlist","trackList","track")
```

value (name1[[,index1][,name2[,index2]][,…],nameN[,indexN])

Получение значения элемента nameN в заданной иерархии. Если index не задан, то берется первый элемент.

Возвращаемое значение: строка

Пример:

```
local v=x:value("playlist","trackList","track",4,"title")
```

attribute (name1[[,index1][,name2[,index2]][,…],nameN[,indexN],attr)

Получение значения атрибута attr элемента nameN в заданной иерархии. Если index не задан, то берется первый элемент.

Возвращаемое значение: строка

Пример:

```
local a=x:attribute("playlist","trackList","track",4,"extension","uri")
```

2.8 ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

util.load(url)

Загрузка документа из указанного источника. В качестве источника может выступать локальный файл или URL адрес (поддерживаются протоколы HTTP,FTP)

Возвращаемое значение: строка

Пример:

```
local d=util.load("ftp://ftp.gnu.org/pub/information.txt")
```

util.encode(text[,charset])

Перекодировка строки в UTF-8 формат. Параметр charset – строка с названием кодировки оригинального текста.

Возвращаемое значение: строка

Пример:

```
local d=util.load("ftp://ftp.gnu.org/pub/information.txt")
local a=util.encode(d,"WINDOWS-1251")
```

util.ls(path)

Получение списка файлов и папок из заданного каталога path. Доступны локальные, сетевые, и FTP каталоги.

Возвращаемое значение: таблица, аналог таблицы списка формируемого плагином

Пример:

```
local d=util.ls("ftp://ftp.gnu.org/")
```

```
for i=1,#d do
   log.i("file "..tostring(d[i].title))
end
```

util.iconv(subject, incharset, outcharset)

Перекодировка строки subject из кодировки incharset в кодировку outcharset. Параметры incharset, outcharset – строка с названием кодировки

Возвращаемое значение: строка

Пример:

```
local d=util.load("ftp://ftp.gnu.org/pub/information.txt")
local a=util.iconv(d,"WINDOWS-1251", "UTF-8")
```

os.filesize(filepath)

Получение размера локального файла

Возвращаемое значение: число

Пример:

```
local size=os.filesize('/sdcard/test.m3u')
```

os.statfs(path)

Получение информации о локальном каталоге/носителе данных

Возвращаемое значение: таблица, параметры: total – всего, used – занято, free – свободно, percent – процент занятого

Пример:

```
local stat=os.statfs('/sdcard')
print('Free '..tostring(stat.free))
```

os.cache(path)

Получение пути к файлу для сохранения данных в кеш

Возвращаемое значение: строка

Пример:

```
local cachefile=os.cache('pl_list.txt')
```