Краткая инструкция по созданию плагина

8.1 Введение

Механизм PHP плагинов позволяет расширить Dune GUI пользовательскими приложениями запрограммированных в языке PHP с применением PHP API Dune HD STB. PHP плагины созданные с помощью этих механизмов хранятся в STB виртуальной памяти и запускаются на STB.

PHP API не предоставляет доступ для построение произвольных TV GUI интерфейсов, и создания определенных обработчиков событий для удаленного администрирования в произвольном порядке.
Взамен представлен высокоуровневый GUI framework с фиксированным набором возможностей, которые могут быть реализованы в произвольном порядке.

Основная цель PHP плагинов в реализации механизмов пользовательских IPTV/VOD приложений, хотя это может быть применено и для других целей.

Возможности обеспеченные механизмами PHP плагинов включают:

- Отображение пользовательских экранов со списками различных объектов(ТВ каналы, видео, категории). Функционал для реализации таких интерфейсов подобный к технологии «dune_folder.txt»
см. http://dune-hd.com/support/misc/dune_folder_howto.txt
- Отображение информации о фильме в определенном экране.
- Отображение разнообразных экранов и диалоговых окон с определенными GUI элементами для пользовательского ввода(поля ввода, кнопки, выпадающие списки и др.).
- Запуск воспроизведения TV каналов, или VOD контента в специальном режиме(со специальными TV или VOD GUI элементами управления).

Преимущественно пользовательские приложения (пользовательские IPTV/VOD приложения) PHP плагинов реализуются по следующих причинах:

- Уменьшение трудозатрат.
  - Не нужно реализовывать GUI с ноля, высокоуровневый GUI framework позволяет сделать всё в короткие сроки и легкую реализацию стандартных IPTV и VOD функциональных элементов.
  - Достаточно предоставить данные, которые GUI тема распределит автоматически наилучшим образом.
- Легкая интеграция с DUNE GUI
  - Визуальная реализация не отличается от других модулей оболочки.
  - Выбор другой темы оболочки автоматически влияет на пользовательское приложение.
- Функционал, который автоматически предоставляется для пользовательских приложений
  - Графическое отображение высокого качества FullHD с разрешением (1920x1080).
  - Высокопроизводительный GUI.
  - Анимационные эффекты.
  - GUI структура разделенная на 4 блока (адресный блок, блок отображение даты и времени с погодными данными, главный блок, опциональная правая панель с отображением деталей).
  - Возможность интеграции пользовательского приложение в разделе

оболочки (Настройки).
- PHP плагин сохраняется STB виртуальной памяти и запускается на устройстве.
- Это позволяет интегрироваться с существующими серверами IPTV по API, без надобности, что-то хранить на стороннем веб-сервере.
- Позволяет уменьшить количество функциональности, которая должна быть обеспеченна хостингом и применяться на веб-серверах.
- Функциональность позволяет задействовать хеширование данных со стороны клиента, что позволяет запрашивать данные другим способом.

8.2 Поддерживаемые STB модели

STB модели базируются на следующих чипсетах, которые поддерживают:

- Sigma Designs 864x
- Sigma Designs 865x
- Sigma Designs 867x

Следующие технологии включают модели STB:

- Dune HD Smart D1/B1/H1
- Dune HD Max
- Dune HD Duo
- Dune HD Lite53D
- Dune HD TV series (TV-101 etc)

8.3 Прошивки поддерживающие технологию

Технология PHP плагинов является новинкой, которая стала доступной с сентября 2011г, распространение которой произошло с появлением новых прошивок. Возможно Вам потребуются загрузить и установить специально разработанную версию прошивки, в которой доступна технология PHP плагинов.

Проверить поддерживает ли Ваша STB приставка технологию PHP плагинов можно, открыв **Настройки → Прочее → Плагины**. Если данного раздела не существует, Вам нужно обновить прошивку на поддерживающую PHP плагины.

Прошивка 2011-09-19 (110919) или более новые, определенно содержат поддержку технологии PHP плагинов.

Также, с тех пор, как механизм PHP и PHP API является в активной разработке (с сентября 2011г.) и находится в не зафиксированном виде, поэтому некоторые версии прошивок могут содержать некоторые различия и особенности в PHP механизмах, которые в свою очередь могут работать или нет в определенных версиях прошивок. Для разработки PHP плагинов рекомендуется использовать последнюю доступную версию прошивки, оснащенную документацией или примерами.

8.4 Установка PHP плагина

Одна STB приставка может включать множество установленных плагинов.

Определенные прошивки могут содержать в себе уже установленные плагины, с возможностью установки новых пользователем.

Для ручной установки PHP плагина применяется специальный файловый «инсталлятор плагина». Инсталлятор представляет собой простой ZIP архив содержащий "dune_plugin.xml" файл расположенный в «корневой» папке и других файлов и папок.

Для инсталлировать PHP плагин достаточно запустить файл «инталлятора плагина», (dune_plugin*.zip) хранящийся на накопителе(флешка, samba папка). Плагин будет установлен во внутри памяти STB приставки.

Отметьте, что по умолчанию STB теребует инициализацию «Хранение во флеш памяти накопителя» (специальный раздел во флеш памяти STB приставки) или инициализацию «системного накопителя» (внешнее устройство применяемое для системных целей), которое будет активировано в процессе ручной установки плагина. Если во время инсталляции еще не один из вариантов не активирован STB будет предлагать активировать «Хранение во флеш памяти накопителя». Для получения больше информации о «Хранение во флеш памяти накопителя» или «Системном накопителе» пожалуйста см. http://dune-hd.com/support/usb_flash_drive

PHP плагины установленные вручную пользователем можно просмотреть **Настройки → Прочее → Плагины** пункте меню. Это меню также предоставляет возможность удалить, ранее установленные плагины(выберите плагин, после чего нажмите Enter, далее выберите пункт меню «Удалить»).

Для переустановки плагина (например, установка более новой версии), первое — удалите предыдущую установленную версию, после чего установите новую.

Предустановленные плагины (добавленные в прошивку) не отображаются в разделе **Настройки → Прочее → Плагины** и не могут быть удалены.

8.5 Файловая структура PHP плагина

PHP плагин представляет собой папку содержащую следующие файлы.

1.) dune_plugin.xml (обязательно)

- Этот файл является конфигурацией в XML формате. Это единственный файл требуемый STB в процессе загрузки плагина. Все другие файлы плагина(если такие объявлены) в конфигурации могут иметь другие имена.

2.) PHP приложение содержит hd_create_plugin() функцию (обязательно) . Путь к файлу содержащему эту функцию объявлен в dune_plugin.xml

3.) Любые другие файлы, для примера, некоторые ресурсы (картинки, иконки) (опционально) . На эти файлы могут быть созданы ссылки благодаря конфигурации в dune_plugin.xml или конфигурации других файлов, или использованы для других целей.

8.6 Настройка файла конфигурации

Пример конфигурации плагина названный «sample» выглядит как представлено далее:

```
<dune_plugin>
 <name>sample</name>
 <caption>Sample</caption>
 <type>php</type>
 <params>
   <program>sample.php</program>
 </params>
 <entry_points>
  <entry_point>
    <parent_media_url>root://tv</parent_media_url>
    <media_url>main_tv</media_url>
    <caption>Sample.TV</caption>
    <icon_url>plugin_file://icons/sample_tv.png</icon_url>
    <actions>
     <key_enter>
      <type>plugin_open_folder</type>
     </key_enter>
     <key_play>
      <type>plugin_tv_play</type>
     </key_play>
    </actions>
    <show_default_value>yes</show_default_value>
    <show_cookie_name>show_tv_entry_point</show_cookie_name>
    <ip_address_required>yes</ip_address_required>
    <valid_time_required>yes</valid_time_required>
  </entry_point>
  <entry_point>
    <parent_media_url>root://applications</parent_media_url>
    <media_url>main_vod</media_url>
    <caption>Sample.VOD</caption>
    <icon_url>plugin_file://icons/sample_vod.png</icon_url>
    <actions>
     <key_enter>
      <type>plugin_open_folder</type>
     </key_enter>
    </actions>
    <show_default_value>yes</show_default_value>
    <show_cookie_name>show_vod_entry_point</show_cookie_name>
    <ip_address_required>yes</ip_address_required>
    <valid_time_required>yes</valid_time_required>
  </entry_point>
 </entry_points>
 <auto_resume>
   <enable>yes</enable>
   <ip_address_required>yes</ip_address_required>
   <valid_time_required>yes</valid_time_required>
 </auto_resume>
</dune_plugin>
```

Детали:

dune_plugin->name — идентификатор плагина. Обязательно должен быть не пустым и содержать английское буквенно-цифровое значение с возможностью добавить ('-', '.', '_') символы.

- dune_plugin->caption: - заголовок применяемый в:
Настройки → Прочее → Плагины

- dune_plugin->type и dune_plugin->params: тип используемого «движка» для выполнения плагина и параметры запуска. На данный момент, поддерживается только «php» тип, имеющий один обязательный параметр 'program' указывающий на главный php файл плагина.

- dune_plugin->entry_points: список точек входа в Dune меню. Каждая dune_plugin->entry_points->entry_point может содержать следующие поля:

- 'parent_media_url': URL пункта меню, где будет размещен ярлык для точки запуска плагина. На данный момент доступны следующие значения пункты меню:
  root://tv
  root://applications
  setup://applications

- 'caption', 'icon_url': визуальные параметры элемента точки запуска.

- 'media_url': идентификатор узла в иерархии меню плагина. Идентификатор должен быть уникальным в масштабе плагина. 'media_url' будет использоваться в некоторых операциях плагина.

- 'actions': event->action маршрутизация. Является списком пар:
ключ действия => спецификация действия. Если точка входа сфокусирована и настроенный ключ является выбранным полученное действие будет выполнено с помощью Dune GUI.

см. раздел «Плагин PHP программа», в котором описано, как Dune GUI работает с точками старта.
см. раздел «GUI действия» - раздел, который описывает более подробно концепцию GUI действий и для получения списка доступных GUI действий.

- 'show_default_value' и 'show_cookie_name': представлено как опция для отображения или скрытия(show/hide) точки старта плагина.

'show_cookie_name' — если кука (для примера, постоянное имя свойства) , и ее значение определено, как логическое. Может принимать возможное решение отображать ли точку запуска в Dune меню или нет. Синтаксис значения следующий:

yes
no
lang(<lang1>,...,<langN>).

'show_default_value' — применяется если 'show_cookie_name' не указано.

- 'ip_address_required': возможные значения: yes или no. Если установлено значение 'yes' и IP адрес еще не определен, тогда открытие точки старта будет причиной для отображение диалога: «Ожидание получение IP адреса», который будет отображаться

до момента его получения, нажатия пользователем кнопки «Отмена» или истечении времени операции.

- 'valid_time_required': возможные значения: yes или no. Если установлено значение 'yes' и текущее время еще не было синхронизировано через интернет, открытие такой точки старта будет причиной для появления диалога «Ожидание синхронизации времени», который будет отображаться до момента его получения, нажатия пользователем кнопки «Отмена» или истечении времени операции.

- auto_resume: спецификация автоматического продолжение, особенность которая отключена по умолчанию. Поддерживаются следующие поля:

  - enable: yes/no. Эта опция включает опцию автоматического продолжения. По умолчанию, все поддерживаемые значения для выбора (живое TV воспроизведение, архивное TV воспроизведение, VOD воспроизведение) можно задействовать.

  - action: GUI действие. Это GUI действие будет выполнено в выборе auto_resume взамен стандартных алгоритмов.

  - ip_address_required: возможные значениния: «yes» или «no». Если это значение установлено «yes» диалог «Ожидание получение IP адреса» будет отображаться если будет нужно в процессе автоматического продолжения.

  - valid_time_required: возможные значениния: «yes» или «no». Если это значение установлено 'yes' диалог «Ожидание синхронизации времени» будет отображаться если будет нужно в процессе автоматического продолжения.

8.7 Обзор высокоуровневого проектного решения

Это является центральным Dune GUI компонентом, который реализует основную работу связанную с: дистанционным управлением, отображением графики, воспроизведении видео потоков подобным образом.

Dune GUI может также управлять множеством PHP плагинов.

Dune GUI запускает отдельные процессы, для которых PHP интерпретатор отдельно инициализируем.

Dune GUI обращается к PHP плагину используя IPC (интер процесс коммуникацию), в отдельных: UNIX каналах. Протокол передачи представлен простым текстом в формате JSON, предназначенного для структурирования данных.

Коммуникация с процессом однонаправленная: Dune GUI создает запрос (или вызов) и асинхронно ожидает ответа от PHP плагина. Если PHP плагин потерпел неудачу и ответ не был получен в определенные сроки, будет выдано сообщение о ошибке.

PHP плагин не может инициализировать запрос к Dune GUI.

В основном, вызов от Dune GUI к PHP плагину может сформулирован по следующим мета объявлениям:

строка *call_plugin(строка call_ctx_json);*

- Dune GUI подготавливает структуру определенного вида, где определены коды операций, параметры операций (которые именуются, как «входные данные»).
- Эта структура является закодированной (сериализация) в строку благодаря использованию JSON, которая может быть обработана обратно в Dune GUI.
- Dune GUI десериализация строки в структуру и использование данных для определенных действий.

Таким образом, в конечном счёте, PHP плагину нужно обеспечить выполнение только одной функции.

строка call_plugin(строка call_ctx_json);
(актуальное имя см. ниже в разделе «Системные файлы PHP»)

Несмотря на это, это также является громоздким и подверженным ошибкам, при реализации похожей логики каждого из плагинов с начала, таким образом SDK обеспечивает оболочку (framework), которая приводит все работы формирования данных в определенный порядок и реализует правильный вызов.

8.8 Системные файлы PHP

STB содержит множество системных файлов PHP, которые автоматически подгружаются из исходников PHP интерпретатором, примерно перед загрузкой PHP программы. PHP определяет и элементы обеспеченные благодаря этим файлам, которые автоматически будут доступны для плагинов PHP программ.

На STB эти файлы располагаются в директории: "/firmware_ext/php". Для удобства, эти файлы также доступны в PHP SDK пакете, в "php_api_definitions" папке.

Эти файлы в основном определяют интерфейсы между Dune GUI и плагином PHP.
Перечень файлов:
- bootstrap.php
    - Настраивает отчет ошибок в рамках PHP интерпретатора таким образом, что все ошибки записываются во внутренний лог файл.
    - Настраивает корректный (текущий) часовой пояс.
    - Обеспечивает следующие глобальные функции:
        - *hd_print($text_message)*

          Эта функция должна быть использована взамен PHP функций 'echo', 'print', 'printf' и подобных.

          Благодаря использованию этой функции разработчик может быть уверен, что текст для лог файла был сформирован в правильном формате.

          Эта функция использует $HD_NEW_LINE контейнер, который является пустой строкой при старте.

        - hd_silence_warnings()

Эта функция должна быть использована для временной блокировки системных PHP предупреждений.

- hd_restore_warnings()

Эта функция восстанавливает запись в лог предупреждений после hd_silence_warnings().

Также ниже описанные функции доступны в bootstrap.php, хотя они обычно не должны быть явно использованы в PHP плагинах:
- hd_error_handler()
- hd_shutdown_handler()
- hd_error_silencer()

- dune_plugin.php
- Определяет высокоуровневый интерфейс между Dune GUI и плагином PHP. DunePlugin интерфейс устанавливает операции, которые Dune GUI может вызвать из PHP плагина.

Если стандартная оболочка (framework)(см. ниже) используется, все вызовы из Dune GUI к плагину PHP являются отправленными в отдельный DunePlugin интерфейс.

- dune_plugin_fw.php
Обеспечивает высокоуровневый интерфейс (в терминологии строк), между Dune GUI и плагином PHP.

Обеспечивает функции утилиты.

- dune_api.php
Содержит определение различных строчных констант, которые помогают в передаче данных между плагином PHP и Dune GUI.

8.9 Простой плагин PHP

Каждый PHP плагин, возможно, должен проделывать следующие шаги:
- Обеспечивать класс наследованный от DunePluginFw;
- Реализовывать DunePluginFw::call_plugin() в производном классе;
- Назначать экземпляр класса для DunePluginFw::$instance;

В настоящее время, DunePlugin интерфейс выглядит, как описано далее:

```
 interface DunePlugin
{
   // PluginFolderView
   public function get_folder_view(
      /* [in]     String                  */  $media_url,
      /* [inout]  Map: Key -> Value       */  &$plugin_cookies);

   // PluginFolderView
   public function get_next_folder_view(
      /* [in]     String                  */  $media_url,
```

```
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // PluginTvInfo
    public function get_tv_info(
        /* [in]     String                     */  $media_url,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // String
    public function get_tv_stream_url(
        /* [in]     String                     */  $playback_url,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // PluginVodInfo
    public function get_vod_info(
        /* [in]     String                     */  $media_url,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // String
    public function get_vod_stream_url(
        /* [in]     String                     */  $media_url,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // PluginRegularFolderRange
    public function get_regular_folder_items(
        /* [in]     String                     */  $media_url,
        /* [in]     int                        */  $from_ndx,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // List<PluginTvEpgProgram>
    public function get_day_epg(
        /* [in]     String                     */  $channel_id,
        /* [in]     int                        */  $day_start_tm_sec,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // String
    public function get_tv_playback_url(
        /* [in]     String                     */  $channel_id,
        /* [in]     int                        */  $archive_tm_sec,
        /* [in]     String                     */  $protect_code,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // void
    public function change_tv_favorites(
        /* [in]     String                     */  $op_type,
        /* [in]     String                     */  $channel_id,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);

    // GuiAction
    public function handle_user_input(
        /* [in]     Map: Key -> Value          */  &$user_input,
        /* [inout]  Map: Key -> Value          */  &$plugin_cookies);
}
```

Тип вывода операции плагина:

```
class PluginOutputData
{
   Boolean                has_data;
   PluginOutputDataType    data_type;
   Object                 data;
   Map<String, String>     plugin_cookies;
   Boolean                is_error;
   GuiAction               error_action;
}
```

Каждая операция плагина должна возвращать один из следующих результатов:
- нормальный вывод: поле «is_error» = «false», «error_action» = «null», «has_data» = «true», «data» содержит нормальный вывод данных, специфицированный в «data_type».
- вывод ошибки: поле «is_error» = «true», «error_action», которое может быть установлено, как определенное GUI действие для выполнения.

8.10 «Жизненный цикл» PHP плагина

Каждый PHP плагин выполняется в отдельном экземпляре PHP интерпретатора (запускается, как отдельный, собственный процесс). Экземпляр PHP интерпретатора для каждого плагина создается при его первом использовании.

PHP плагин всегда обрабатывает только на один вызов полученный из Dune GUI за один раз. Со сложной структурой запросы из Dune GUI к плагину PHP всегда подвержены преобразовываются.

Как правило, последовательные запросы из Dune GUI к PHP плагину отправляются одному и тому же экземпляру PHP интерпретатора. Поэтому глобальные переменные являются защищенными между последовательными вызовами.

Тем не менее, каждый экземпляр PHP интерпретатора может быть уничтожен или создан по новой в любой момент(между вызовами Dune GUI к PHP плагину), когда это нужно для Dune GUI. Когда это случается глобальные переменные не сохраняются.

Жизненный цикл PHP плагина не предполагает, что глобальные переменные будут всегда защищенными между последовательными запросами от Dune GUI к PHP плагину. PHP плагин может использовать глобальные переменные, к примеру, для реализации хеширования(может быть полезным с соображений производительности), но нужно быть подготовленным к тому, что значения глобальных переменных могут пропасть между последовательными запросами, что требует реализацию присущей стратегии для повторного заполнения хеша.

Если это возможно, PHP плагин должен быть спроектирован, как автономное приложение. Если требуется некоторое преходящее глобальное состояние и хранение этого состояния критично между последовательными запросами к PHP плагину, этот вопрос должен реализоваться следующими специальными путями:
- Состояние должно быть закодированным в параметрах, которые передаются между PHP плагином и Dune GUI, таким образом Dune GUI хранит состояние.
- Состояние должно быть сохранено в разделе для временного хранения данных

(RAM в STB).
- Состояние должно быть сохранено на интернет сервере (если плагин взаимодействует с интернет сервером).

8.11 Программная среда PHP плагина

1.) Куки

- каждый плагин имеет список постоянных строк, названных, как куки. Они отображаются, как (string => string) и доступны благодаря ссылкам для каждой операции PHP плагина. Таким образом операции могут использовать значение кук и также их модифицировать. Куки содержат значения между операциями и даже передаются после процесса перезагрузки STB.

ОТМЕТЬТЕ: куки не поддерживают хранение данных больших объемов, что может быть причиной не успешного запуска PHP плагина при перезапуске.

2.) Системные свойства

- PHP плагин имеет доступ к массиву DuneSystem::$properties. На данный момент доступны следующие поддерживаются следующие свойства:

'install_dir_path' — путь к директории куда установлен плагин;
'tmp_dir_path' — путь к директории, после /tmp, которая предназначена для хранения временных данных плагина. Dune GUI гарантирует, что папка существует.
'data_dir_path' — путь к директории для хранения постоянных данных. ОТМЕТЬТЕ: что директория не доступна без инициализированного «системного накопителя» или «хранения во флеш памяти».

ОТМЕТЬТЕ: После каждой записи в постоянную память, должна быть немедленно выполнены функция «sync» или системные вызовы.

8.12 Обзор основных особенностей PHP плагина

Каждый метод в DunePlugin интерфейсе соответствует таким образом вызванной операции. Каждая операция имеет тип ввода (простой) и тип вывода (определенного значения комплекс). Данные вывода должны быть представлены, как метод DunePlugin возвращающий PHP массив. Ключи для массива вывода объявлены в системном файле dune_api.php.

PHP плагин может расширять базовую функциональность Dune в следующих способы:
- расширение Dune меню иерархии (Dune меню также иногда именуют, как «папки») Некоторые основные Dune папки (в данном случае: ТВ, Приложения, Настройки) могут быть расширены благодаря добавлению новых элементов, которые названы «точками старта плагина». Каждая точка старта принадлежит плагину. Как правило, точки старта содержат иерархию папок управляемую плагином.

- Запуск ТВ воспроизведения. Воспроизведение плагином ТВ, является специальным режимом воспроизведения, предоставляющего доступ к запуску каналов, отображению списка каналов объеденных в категории (также именуемых как группы), просмотр EGP.

- Запуск VOD воспроизведения. Воспроизведения плагином VOD, является

специальным режимом воспроизведения, предоставляющего доступ к проигрыванию фильмов или серий (представляет собой список эпизодов объеденных по определенному заголовку).

- Отображение по-умолчанию и основное поведение папок Dune расширено благодаря плагинам следующим способом:
    - новый элемент добавлен для каждого зарегистрированного плагина и каждой точки старта в конфигурации плагина.
    - отдельный плагин получает управление над вводом пользователя, когда точка старта была выбрана. На данный момент плагин может переопределять стандартное поведение для следующих управляющих кнопок в основной папке: ENTER, PLAY.

Техника примененная плагинами влияет на поведение Dune GUI, который является базовым решением Dune GUI операций. Для большей информации о GUI действиях см. раздел «GUI действия».

Кратко, GUI действие это инструкция для Dune GUI, что нужно делать, когда пользователь нажимает кнопку пульта в определенных состояниях GUI. GUI действия, как правило, создаются из PHP кода, в виде массива с следующими полями:
- handler_string_id : [строка] тип действия
- data: [массив PHP] массив параметров настраивающий действие

GUI действия могут быть таких типов:
- PLUGIN_OPEN_FOLDER. Отображает внутри некоторый GUI элемент. Открывает новую подпапку (меню) полностью управляется плагином. Такие папки названы «папками плагина».
- PLUGIN_TV_PLAY. Запускает ТВ вопроизведение.
- PLUGIN_VOD_PLAY. Запускает VOD вопроизведение.

Также существует одно специальное действие, названое HANDLE_USER_INPUT. Выполение HANDLE_USER_INPUT включает:
- запуск некоторого произвольного кода плагина. Конкретнее, операцию плагина handle_user_input() выполнив которую, получает текущее Dune GUI состояния, как параметр.
- немедленное выполнение другого GUI действия определенный плагином. Фактически возвращаемое значение функцией handle_user_input() представлено как выполнение действий GUI.

Папка плагина имеет строковый идентификатор, который явно не был вызван 'media_url'. Если обобщить, каждая папка (включая точки старта) имеет уникальное 'media_url' значение в масштабе одного плагина.

Папки плагинов делятся на три основных типа:

1.) Регулярные папки

    Более всего применяемый тип папок для отображения списков объектов любого рода, как таблицы с фиксированным количеством строк и колонок. Каждая ячейка таблицы может содержать картинку или текст. Визуальное отображение каждой из регулярных папок может быть упорядочено благодаря многочисленным настройкам. Также плагин может определять многочисленное количество вариантов визуальных

представлений. Эти варианты могут переключаться с помощью кнопки пульта A_RED. Давайте рассмотрим вариант представления регулярной папки.

Регулярная папка плагина ведет себя похожим образом, как и основные папки Dune, но эффекты нажатия на кнопки пульта должны быть явно определены плагином, в других случаях они будет проигнорированы. Кнопки пульта, которые можно применять: ENTER, PLAY, INFO, POPUP_MENU, B_GREEN, C_YELLOW, D_BLUE. Другими словами плагин может назначать GUI действие, к прослушиваемым кнопкам.

Регулярные папки могут быть разделены на:
- простая (загруженная) — все содержания папки загружается один раз.
- с ленивой подгрузкой где известен размер: число элементов папки, которое известно заранее, применяется для большого числа объектов, где объекты подгружаются порциями, когда пользователь перематывает контент.
- с ленивой подгрузкой, где не известно количество всех элементов: аналогично предыдущей, но количество всех элементов неизвестно в процессе извлечения.

2.) Папка элементов

Тип папки сформирован для применения в настройках. Папка элементов выглядит, как вертикальный список GUI элементов и текстовых констант.

Следующие элементы поддерживаются:
- кнопка
- выпадающий список
- текстовое поле

Плагин может назначать специфические GUI действия для следующих событий:
- нажатие на кнопку
- значение выпадающего списка было изменено
- текстовое поле было изменено

HANDLE_USER_INPUT в GUI должен быть использован для фиксации и сохранения изменений.

3.) Папка подробностей видео

Специальный слой представленный детальной информацией о фильме. Следующие свойства фильма которые могут быть отображены в таких папках:
- название фильма
- оригинальное название
- постер
- жанр
- страна
- год

- режиссёр
- актёры
и другие

Слой также содержит одну или две кнопки:
- Обязательное отображение кнопки «Просмор», в левом нижнем углу с предопределенными обработчиками. Количество серий влияют на обработчик следующим образом:

  - (number_of_series > 1) => открывается папка с сериями.
  - (number_of_series = 1) => VOD проигрывание запущенное для видео.
- Опциональная кнопка с пользовательским названием, расположенная в правом нижнем углу. Эта кнопка может быть, по выбору задействована для «Добавление в избранное» и «Удаления из избранного» или для других целей.

Для поддержки архитектуры отображения папок плагин PHP должен реализовать метод: DunePlugin::get_folder_view($media_url).

Для поддержки отображения разных представлений (нажатие A_RED) PHP плагин должен реализовать метод:
DunePlugin::get_next_folder_view($media_url).

Для поддержки «ленивой загрузки» элементов регулярных папок плагин должен реализовать метод: DunePlugin::get_regular_folder_items($media_url) последовательно с DunePlugin::get_folder_view($media_url) .
Для фиксации/сохранения изменений в настройках плагин должен реализовать метод: DunePlugin::handle_user_input($params).

ТВ проигрывание имеет опциональные особенности:
- EPG отображение
- Проигрывание архива
- Защищенные каналы
и другие.

Оба ТВ и VOD режима проигрывания имеют опциональные особенности:
- переменная URL потока или видео

Минимальные требования для поддержки ТВ проигрывания, плагин PHP должен реализовать методы:
DunePlugin::get_tv_info($media_url)
DunePlugin::get_tv_playback_url($media_url, $archive_tm_sec, $protect_code).

Для поддержки EPG отображения плагин должен реализовать метод:
DunePlugin::get_day_epg($channel_id, $day_start_tm_sec);

Для проигрывания архива DunePlugin::get_tv_playback_url(...) должно быть не пустым поле класса проигрывателя $archive_tm_sec (последовательно с возвращенными данными с get_tv_info())

Для особенности поддержки защищенных каналов DunePlugin::get_tv_playback_url(...)

должен может быть взято значение аргумента $protect_code (для формирование URL).

Для поддержки проигрывания возможных URL ТВ потоков, PHP плагин должен реализовать метод: DunePlugin::get_tv_stream_url($playback_url).

Для поддержки проигрывания возможных URL VOD потоков, PHP плагин должен реализовать метод:
DunePlugin::get_vod_stream_url($playback_url).

8.13 Типичные записи проекта PHP плагина

1.) Файл конфигурации содержит следующее:

- имя плагина
- заголовок плагина
- тип плагина (php) и главный PHP файл (plugin_name>.php)
- список точек старта
- настройку автоматического продолжения

Если плагин обеспечивает IPTV сервис, тогда нужно добавить точку старта в ТВ раздел меню. Эта точка старта реагирует, как PLUGIN_OPEN_FOLDER при нажатии кнопки пульта ENTER и PLUGIN_TV_PLAY при нажатии PLAY.

Если плагин обеспечивает не IPTV сервисы, тогда можно добавить одну или больше точек старта в пункт меню «Приложения».

Также типичный плагина добавляет точку старта в пункте меню **Настройк → Приложения**.

Если плагин обеспечивает воспроизведение видео/аудио потоков, он всегда может обращаться к возможности «авто продолжения», которая позволяет автоматически продолжать воспроизведение с определенного состояния после перезапуска STB.

2. Иерархия папок

Обычно ТВ точка старта содержит 2 уровня «регулярных папок»:

- первый уровень содержит список категорий ТВ каналов. Каждый элемент категория действует, как папка, когда нажата кнопка ENTER. ТВ воспроизведение запускается при нажатии PLAY.
- второй уровень содержит каналы. Каждый элемент канал действует, как TV_PLAY при нажатиях кнопок пульта ENTER и PLAY.

Точки старта обеспечивают доступ к VOD сервисам, которые могут иметь произвольную глубину вложений категорий базы данных фильмов. Каждая категория представлена, как регулярная папка имеющая подкатегории и фильмы, как элементы. Элемент видео открывает папку или проигрывание. Проигрывание видео может быть запущено изнутри папки фильмов.

3. Проигрывание

Типичное ТВ проигрывание запускается:

- при нажатии ENTER/PLAY кнопок пульта в регулярной папке на элементе представляющем канал.
- при нажатии PLAY в регулярной папке на элементе представляющем категорию канала.
- при нажатии на PLAY на точке старта.

Типичное VOD проигрывание запускается:
- при нажатии ENTER/PLAY кнопок пульта в регулярной папке на элементе представляющем видео или серию видео.
- при нажатии ENTER/PLAY в видео папке.

Также TB/VOD проигрывание может быть запущено автоматически на STB при нажатии на кнопку POWER в режиме «авто продолжения».

4. Экран настроек

Точка старта экрана настроек располагается в меню **Настройки → Приложения**. Экран настроек реализован подобным образом, как тип «Экран элементов».

Как правило, экран настроек содержит отображение в главном окне выпадающих списков со значениями «Да», «Нет».

Действие HANDLE_USER_INPUT GUI и обработчик handle_user_input() должны быть применены для фиксации изменений и сохранения настроек в определенном месте.
Как правило, куки плагина используются для хранения локальных настроек плагина.

8.14 Операции PHP плагина

Ниже представлено краткое описание операций плагина.

ОТМЕТЬТЕ: см. dune_api.php для того чтобы получить полный список полей для каждого возвращаемого типа данных.

1. Get_folder_view()

```
// PluginFolderView
public function get_folder_view(
  /* [in]     String                      */  $media_url,
  /* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Возвращает текущую папку для отображения данных в меню плагина с идентифицируемом по $media_url.

Эта операция вызывается, когда Dune GUI выполняет действиее PLUGIN_OPEN_FOLDER GUI.

Возвращаемые данные:

```
class PluginFolderView
{
    PluginFolderViewKind         view_kind;
    union (PluginFolderViewKind) data;
    Boolean                      multiple_views_supported;
```

```
    PluginArchiveDef            archive;
}
```

Структура является опциональной спецификацией использованной для этой папки. см. раздел «Разнообразные особенности» для получения большей информации о спецификациях.

Поддерживается множество сконфигурированных «отображение множественных видов», если это доступно для этого типа папки. Для большей информации см. описание для операции плагина GET_NEXT_FOLDER_VIEW.

Вид отображения (view_kind) может быть следующим:
- PLUGIN_FOLDER_VIEW_REGULAR
- PLUGIN_FOLDER_VIEW_CONTROLS
- PLUGIN_FOLDER_VIEW_MOVIE

Данные (data) специфически указываются в зависимости от (view_kind):
- регулярные папки: PluginRegularFolderView
- папка элементов: PluginControlsFolderView
- папки видео: PluginMovieFolderView

Регулярные папки

```
class PluginRegularFolderView
{
    ViewParams               view_params;
    ViewItemParams           base_view_item_params;
    ViewItemParams           not_loaded_view_item_params;
    GuiActionMap             actions;
    Boolean                  async_icon_loading;
    PluginRegularFolderRange initial_range;
}
```

Поле 'initial_range' указывает элементы инициализации папки.
При простом отображении, все элементы должны быть загружены в 'initial_range'.
При отображении с «ленивой продгрузкой» поле 'initial_range' папки должно содержать первую порцию данных или может быть оставлена пустым.

Поле 'view_params' может настаивать отображение целой папки, с помощью таких параметров, как: num_cols, num_rows, async_icon_loading и др.

Поле 'async_icon_loading' может настраивать вариант загрузки постеров в асинхронном режиме, для более подробной информации см. раздел «Разнообразные особенности»

Поле 'base_view_item_params' может настраивать параметры отображения по-умолчанию для каждого отдельного элемента папки.

Поле 'not_loaded_view_item_params' может настраивать отображение для специальных состояний для каждого отдельного элемента папки. Это используется, когда

1.) элемент представлен еще не загруженной картинкой или 2.) или загруженной с ошибкой (загрузка была не успешной).
ОТМЕТЬТЕ: как правило это значение игнорируется если ViewParams::async_icon_loading == false.

Действия оформляются в таком формате {действие -> GUI действие}.

Папки элементов

```
class PluginControlsFolderView
{
   Array<GuiControlDef>    defs;
   int                     initial_sel_ndx;
}
```

Поле 'defs' представляет оформленный список GUI элементов. Для более подробной информации см. раздел «GUI элементы».

Поле 'initial_sel_ndx' указывает стартовый индекс из GUI элементов, который будет представлен первым. Значение -1 означает, что будет взят первый элемент массива в качестве стартового.

Папка деталей видео

```
class PluginMovieFolderView
{
    PluginMovie     movie;

    Boolean         has_right_button;
    String          right_button_caption;
    GuiAction       right_button_action;

    Boolean         has_multiple_series;
    Boolean         series_media_url;
}
```

Поле 'movie' настраивает многочисленные параметры видео. См. список видео параметров ниже.

Поле 'has_right_button' указывает отобрать или нет кнопку в правом нижнем углу экрана.

Поле 'right_button_caption' настраивает заголовок правой кнопки.

Поле 'right_button_action' указывает GUI действие назначенное правой кнопке.

Поле 'has_multiple_series' настраивает поведение при нажатии на кнопку пульта ENTER при позиционировании на экранной кнопке «Просмотр» в левом нижнем углу, экрана может быть следующим:

- когда FALSE => PLUGIN_VOD_PLAY GUI действие выполненное.
- когда TRUE => PLUGIN_OPEN_FOLDER GUI действие выполненное с применением media_url взятым из поля 'series_media_url'.

Эти схемы спроектированы для реализации пользовательского выбора серий или фильма разделенного на эпизоды. В этом случае нажатие на ENTER кнопки пульта отображает список серий.

Поле 'series_media_url' применяется с 'has_multiple_series' = true. Оно идентифицирует папку, которая должна быть открыта когда пользователь нажимает на кнопку пульта ENTER при позиционировании на экранной кнопке «Просмотр» в левом нижнем углу, экрана.

```
class PluginMovie
{
   String  name;
   String  name_original;
   String  description;
   String  poster_url;
   int     length_min;
   int     year;
   String  directors_str;
   String  scenarios_str;
   String  actors_str;
   String  genres_str;
   String  rate_imdb;
   String  rate_kinopoisk;
   String  rate_mpaa;
   String  country;
   String  budget;
}
```

2. Get_next_folder_view()

```
// PluginFolderView
public function get_next_folder_view(
  /* [in]     String                      */  $media_url,
  /* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Возвращает следующее отображение вида для папки плагина идентифицированную через $media_url.

Некоторые регулярные папки могут иметь несколько видов отображений (например, разное количеством рядов и колонок или разный размер иконок и др.). Такие виды могут меняться благодаря нажатию кнопки пульта A_RED. Для включения этого свойства для отдельной папки плагин должен указывать: PluginFolderView::multiple_views_supported == true. В этом случае нажатие на A_RED будет причиной для вызова функции Get_next_folder_view();

3. Get_regular_folder_items()

```
// PluginRegularFolderRange
public function get_regular_folder_items(
  /* [in]     String                      */  $media_url,
  /* [in]     int                         */  $from_ndx,
  /* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Возвращает специфическую порцию данных элементов папки идентифицированной лагодаря $media_url.

Эта функция является частью особенности «ленивой загрузки» для регулярных папок.

Dune GUI никогда не вызывает эту функцию для простой формы, когда загружены все элементы.

Функция Get_regular_folder_items() вызывается для «ленивой загрузки» каждый раз когда запрошенные данные для отображения еще не загружены (типично когда пользователь перематывает список элементов). Элементы папки хешируются в памяти до момента выхода из папки.

см. Get_folder_view() описание функции для получения подробных деталей.

4. Get_tv_info()

```
// PluginTvInfo
public function get_tv_info(
  /* [in]     String                    */  $media_url,
  /* [inout]  Map: Key -> Value         */  &$plugin_cookies);
```

Возвращает данные нужные для запуска ТВ воспроизведения. Аргумент $media_url используется, как подсказка какой канал должен быть проигран первый.

Операция вызывается, когда Dune GUI выполняет PLUGIN_TV_PLAY GUI действие.

Вывод данных для get_tv_info:

```
class PluginTvInfo
{
   unixtime_t              server_time;
   List<PluginTvGroup>     groups;
   List<PluginTvChannel>   channels;
   Boolean                 show_group_channels_only;
   Boolean                 favorites_supported;
   List<String>            favorite_channel_ids;
   String                  favorites_icon_url;
   String                  initial_group_id;
   String                  initial_channel_id;
   Boolean                 initial_favorites;
   unixtime_t              initial_archive_tm;
   Boolean                 ip_address_required;
   Boolean                 valid_time_required;
   PluginFontSize          epg_font_size;
   PluginArchiveDef        archive;
}
```

Поле 'server_time' указывает текущее время сервера, которое будет использовано при ТВ просмотре. Локальное время применяться только в случае если время сервера отрицательное.

Поля 'groups' и 'channels' указывают список каналов и список групп каналов. см. описание групп и каналов ниже.

Поле show_group_channels_only' указывает отображение каналов только в группах.

Панель ТВ при проигрывании (OSD) содержит три колонки:

- первая колонка содержит список групп
- вторая колонка содержит список каналов выбранных из группы
- третья колонка содержит EPG программу для выбранного канала на определенный день

- если ['show_group_channels_only' == false] вторая колонка содержит все каналы отсортированные по группам не зависимо от того какая категория сейчас выбрана. Этот режим позволяет реализовать взаимосвязь 1 к N между группами и каналами, например каждый канал должен принадлежать к определенной группе. Выбор группы в первой колонке автоматически подставляется, когда пользователь перематывает каналы во второй колонке.

- если ['show_group_channels_only' == true] вторая колонка содержит только каналы выбранные из группы. Этот режим поддерживает взаимосвязи N к N между группами и каналами: канал может быть включен внутри множества групп. Как следствие, пользователь может перемещаться между второй и первой колонками, благодаря нажатиям LEFT/RIGHT, для выбора канала из другой группы.

Поле 'favorites_supported' подключает свойство избранное содержащее:
- добавляется специальная группа «Избранное».
- отображается специальный индикатор возле избранного канала. В частном случае Dune GUI вызывает Change_tv_favorites() функцию, когда пользователь нажимает одну из кнопок пульта D_BLUE, B_GREEN, C_YELLOW. Реализация функции Change_tv_favorites() должна записывать список избранного в постоянное хранилище(например это могу быть куки).

Поле 'favorite_channel_ids' определяет список (массив id) избранных каналов.

Поля 'initial_group_id', 'initial_channel_id', 'initial_favorites' и 'initial_archive_tm' настраивают начальное состояние проигрывания и выбора. Все эти поля опциональные. Если значение поля не установлено, оно будет подобрано автоматически.

Если установлено значение «true» для поля 'initial_archive_tm', появляется возможность архивного проигрывания, где начало старта берется от определенной временной метки (GMT unixtime).

Поля 'ip_address_required' и 'valid_time_required' настраивают дополнительные требования для ТВ проигрывания. Для получения больше деталей см. описание похожих полей в настройках конфигурации плагина. В данном случае, это требования обычно применяются если для некоторых плагинов нужно правильное текущее время для отображения групп, каналов в Dune меню, которые требуют обязательно правильное текущее время.

Каждая группа имеет следующие свойства:

```
class PluginTvGroup
{
  String id;
  String caption;
  String icon_url;
}
```

Поле 'id' указывает идентификатор группы.

Поля 'caption' и 'icon_url' настраивают отображение группы.

Каждый канал имеет следующие свойства:

```
class PluginTvChannel
{
  String          id;
  String          caption;
  List<String>    group_ids;
  String          icon_url;
  int             number;
  Boolean         have_archive;
  Boolean         is_protected;
  Boolean         recording_enabled;
  int             buffering_ms;
  int             timeshift_hours;
  int             past_epg_days;
  int             future_epg_days;
  int             archive_past_sec;
  int             archive_delay_sec;
  Boolean         playback_url_is_stream_url;
}
```

Поле 'id' указывает идентификатор канала.

Поля 'caption' и 'icon_url' настраивают визуальное отображение канала.

Поле 'group_ids' указывает группы к которым принадлежит канал. По крайней мере одна из групп должна быть обязательно указана. Некоторые плагины реализовывают некоторые каналы, которые принадлежат к разным группам, а некоторые нет. См. описание PluginTvInfo::show_group_channels_only для получения более подробных деталей.

Поле 'number' назначает физический номер ТВ канала. Если поле <= 0 => тогда номер будет назначен автоматически первым доступным числом большим ноля.

Поле 'have_archive' включает свойство проигрывания из архива для этого канала. Если поле включено, тогда потребуется реализация фукнкции Get_tv_playback_url(). См. раздел «Разнообразные особенности» для получения подробной информации о проигрывании архива ТВ.

Поле 'is_protected' отмечает этот канал как защищенный, например ограниченный для пользователей, которые не знают секретный код. Если плагин содержит некоторые защищенные каналы, потребуется реализация функции Get_tv_playback_url().

Поле 'recording_enabled' указывает, что для этого канала доступна запись.

Поле 'buffering_ms' указывает время буферизации для проигрывания (1000 = 1сек.).

Поле 'timeshift_hours' устанавливает смещение времени. (еще не реализовано)

Поля 'past_epg_days' и 'future_epg_days' указывают лимит скролирования EPG программы. По-умолчанию установлено 14.

Поле 'archive_past_sec' указывает максимальный временный интервал с текущего момента до максимального доступного элемента прошедшего времени. См. раздел «Различные особенности» для получения больше информации о архивном проигрывании ТВ.

Поле 'archive_delay_sec' указывает интервал времени между текущим моментом и ближайшим моментом в прошлом для которого доступен архив. Для получения подробной информации о переменных потока см. раздел «Разнообразные особенности».

5. Get_day_epg()

```
// List<PluginTvEpgProgram>
public function get_day_epg(
/* [in]     String                      */  $channel_id,
/* [in]     int                         */  $day_start_tm_sec,
/* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Возвращает список EPG структур программ для данного канала и данного дня.

День указан в формате unixtime начинающегося с (0:00)
$day_start_tm_sec всегда делится на 86400.

Вывод должен быть (с возможностью быть пустым) списком структур отсортированных по времени, в порядку по возрастанию.

```
class PluginTvEpgProgram
{
  unixtime_t start_tm_sec;
  unixtime_t end_tm_sec;
  String     name;
  String     description;
}
```

Поле 'start_tm_sec' и 'end_tm_sec' являются стартом и завершением программы в формате.
Поле 'end_tm_sec' является опциональным (значение -1 означает отключенное состояние).
Поле 'name' является обязательным, и указывает имя программы.
Поле 'description' опциональное поле, указывает описание к программе.

6. Change_tv_favorites()

```
// GuiAction
public function change_tv_favorites(
/* [in]     String                      */  $op_type,
/* [in]     String                      */  $channel_id,
/* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Обновляет список избранных ТВ каналов.

Вызывается из режима ТВ проигрывания, когда пользователь нажимает B_GREEN, C_YELLOW, D_BLUE кнопки пульта.

Возвращенное значение является опциональным GUI действием, которое будет выполнено незамедлительно.

Поддерживаются следующие операции обновления

- PLUGIN_FAVORITES_OP_ADD
- PLUGIN_FAVORITES_OP_REMOVE
- PLUGIN_FAVORITES_OP_MOVE_UP
- PLUGIN_FAVORITES_OP_MOVE_DOWN

Реализация change_tv_favorites() должна обновлять собственные постоянные копии избранного.

7. Get_tv_playback_url()

```
// String
public function get_tv_playback_url(
      /* [in]     String                      */  $channel_id,
      /* [in]     int                         */  $archive_tm_sec,
      /* [in]     String                      */  $protect_code,
      /* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Возвращает URL проигрывания для данного канала как строку.

Эта операция вызывается для получения URL проигрывания
- живое проигрывание если $archive_tm_sec <= 0;
- архивное проигрывание если $archive_tm_sec > 0.
См. раздел «Разнообразные особенности» для получения информации о архивном проигрывании.

Непустой защищенный код может быть указан для защищенных каналов.См. раздел «Разнообразные особенности» для получения информации о архивном проигрывании.

Эта операция вызывается каждый раз когда стартует или продолжается проигрывание отдельного канала.

Отметьте: некоторые каналы требуют выполнения дополнительного вызова Get_tv_stream_url() для получения актуального URL потока содержащего URL потока для проигрывания, как параметр. Это, как вариант использования при PluginTvChannel::playback_url_is_stream_url == false.

8. Get_tv_stream_url()

```
// String
public function get_tv_stream_url(
      /* [in]     String                      */  $playback_url,
      /* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Возвращает ТВ URL поток, который имеет ТВ проигрывание URL, как ввод.

Пример возможных ТВ URL потоков:

- Мультикаст TS-over-UDP поток (raw-UDP или RTP-over-UDP):
  - udp:/@ip-address:port
- Юникаст TS-over-HTTP поток:
  - http://ts://host[:port][/path]

Причина для разделения в терминологии 'URL проигрывания' и 'поток URL' является следующей. Иногда плагин не в состоянии обеспечить фиксированным URL для IPTV поточного проигрывания и может только обеспечить путь для получения этого в значении удаленного вызова. Каждый вызов будет возвращать разные результаты. Реализовать такой плагины следует проделав следующее:
- для каждого такого канала PluginTvChannel::playback_url_is_stream_url должно быть установлено в false.
- Get_tv_playback_url() должен вернуть строку которая:
1.) Будет использована для инициализации проигрывания по умолчанию. Также она должна содержать корректный префикс (например: http://ts://);
2.) Будет предоставлена для операции Get_tv_stream_url(), как отдельный параметр.

- Get_tv_stream_url() должна быть реализована для декодирования URL проигрывания, выполняет удаленный вызов, вставляет результаты и возвращает URL актуального IPTV потока.

Вызывать 'URL проигрывание' можно, когда результат Get_tv_stream_url(), используется для актуального проигрывания.

Отметьте: Если PluginTvChannel::playback_url_is_stream_url установлен, как TRUE, тогда URL проигрывание интерпретируется, как URL поток и Get_tv_stream_url() не будет никогда вызвана.

9. Get_vod_info()

```
// PluginVodInfo
public function get_vod_info(
      /* [in]     String                      */  $media_url,
      /* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Возвращает данные для запуска плагина VOD проигрывания используя полученную опцию $media_url, как подсказку с какой из серий начинать.

Эта операция обычно вызывается, когда пользователь нажимает PLAY или ENTER на пульте дистанционного управления в определенной папке плагина имеющей с установленным обработчиком событий 'plugin_vod_play'.

Эта операция запускает проигрывание из списка видео серий.
Когда поле series >= 1.

Вывод данных get_vod_info:

```
class PluginVodInfo
{
```

```
      String                      name;
      String                      description;
      String                      poster_url;
      List<PluginVodSeriesInfo>   series;
      int                         initial_series_ndx;
      int                         buffering_ms;
      Boolean                     ip_address_required;
      Boolean                     valid_time_required;
}

class PluginVodSeriesInfo
{
     String  name;
     String  playback_url;
     Boolean playback_url_is_stream_url;
}
```

Поля 'name', 'description', 'poster_url' и 'series[i]->name' используются для OSD представления.

Поле 'series' указывает список серий для проигрывания. Их форма это определенный вид проигрывания списков.

Поле 'initial_series_ndx' указывает индекс из серий для старта проигрывания.

Поля 'ip_address_required' и 'valid_time_required' указывают корректные поведение в случае некорректного IP адреса и/или проверки времени.
См. описание данных вывода GET_TV_INFO операции для получения более подробной информации.

Поле 'buffering_ms' устанавливает время буферизации в миллисекундах.
Отметьте: На данный момент не реализовано.

Поле 'series->playback_url' указывает URL проигрывания для определенных серий.

Поле 'series->playback_url_is_stream_url':
- true => проигрывание URL должно быть представлено, как видеопоток.
- false => Get_vod_stream_url() операция будет вызвана для получения URL видеопотока, каждое время, когда это нужно.

См. описания GET_TV_STREAM_URL и GET_VOD_STREAM_URL для получения деталей.

10. Get_vod_stream_url()

```
// String
public function get_vod_stream_url(
      /* [in]     String                    */  $playback_url,
      /* [inout]  Map: Key -> Value         */  &$plugin_cookies);
```

Возвращает VOD URL поток имеющий VOD проигрывание, как ввод.

Примеры возможных VOD URL потоков:

- MP4 контейнер через HTTP (более из всех рекомендуемый)
  - http://mp4://host[:port][/path]

- TS контейнер через HTTP
  - http://ts://host[:port][/path]

- Другие контейнеры, через HTTP (поддержка/выполнение ограничено)
  - http://host[:port][/path]

- MMS(поддержка ограничена)
  - mms://host[/path]

Эта операция вызывается каждый раз, когда стартует или продолжено проигрывание и если установлено
Если PluginVodInfo::series[ndx]->playback_url_is_stream_url, как false.
PluginVodInfo::series[ndx]->playback_url применяется, как параметр.
См. описание функции Get_tv_stream_url() для получения больше деталей.

11. Handle_user_input()

```
// GuiAction
public function handle_user_input(
/* [in]     Map: Key -> Value           */  &$user_input,
/* [inout]  Map: Key -> Value           */  &$plugin_cookies);
```

Эта операция вызывается в процессе выполнении HANDLE_USER_INPUT GUI действия.

Функция Handle_user_input() может возвращать другие GuiAction объекты, которые будут незамедлительно выполнены.

Эта операция обеспечивает реализацию множества комплексных претендентов. Вот некоторые из них:

- регулярная папка с элементами разных типов. HANDLE_USER_INPUT GUI действие назначено для определенного ключа в определении таких папок.
- экран настроек с возможностью редактировать список опций. HANDLE_USER_INPUT указывает подтверждение/применение действия для текстового компонента или выпадающего списка или может быть назначено для некоторой кнопки «Сохранить».
- реализация «Всплывающих диалогов» гибкие диалоги с пользовательским списком компонентов. Поведение элементов диалога очень близко к поведению элементов управления папок.

GUI действия

Концепция GUI действия очень важная часть в PHP API плагина. Это применяется плагином для определения поведения Dune GUI в множестве прецедентов.

В частности GUI действия часто назначаются для событий нажатий кнопок пульта управления пользователем. Также GUI действия могут быть вызваны, как результат выполнения других GUI действий. Также GUI действия могут быть возвращены для всех операций взамен стандартному выводу при возникновении ошибки.

Каждое GUI действие подразумевает некоторый алгоритм для выполнения через Dune GUI.

GUI действие полученное из плагина представлено в форме GuiAction:

```
class GuiAction
{
    String              handler_string_id;
    Any                 data;
    Map<String, String> params;
}
```

Поле 'handler_string_id' указывает строку определяющую тип действия. Каждый тип действие будет описан ниже.

Поле 'data' определяет параметры действия. Поле 'data' представлен объектом класса указывающего тип действия. Данные классов для всех поддерживаемых типов действий будет описаны ниже.

Поле 'params' указывает дополнительные параметры действия в строковом типе. Эти параметры автоматически заполненные с определенным видом «выполняемого контекста» в частных случаях прецедентов описанных ниже.

GUI создается для выполнения Dune GUI когда возникает определенное событие.
Выполнение GUI действия включает следующее:
1.) Первый исполнитель создает объект. Разные исполнители будут использовать разные состояния Dune GUI. Исполнитель знает, как выполнить множество типов GUI действий. В настоящее время есть 2 типа исполнителей:

- Главный исполнитель. Он применяется когда мы просматриваем dune меню (иерархию папок) содержащих PopUp меню и диалоги. Этот исполнитель способен выполнить все типы действий.
- Легкий исполнитель. Он применяется в процессе присоединения TV/VOD проигрывания.
  На данный момент этот исполнитель распознает SHOW_ERROR и INVALIDATE_FOLDERS типы действий.

Отметьте: количество исполнителей и множество поддерживаемых ими

действий может быть изменено в будущем.

2.) Исполнитель анализирует тип действия. Если тип неизвестный для исполнителя тогда исполнение прекращается со статусом 'Действие не распознано'. Фактически не распознанные действия являются также обрабатываемыми. В противном случае мы продолжаем с шага (3).

3.) Исполнитель выполняет текущее выполнение с полученного действия. Как результат этого шага следующие данные производятся:
- [int] статус выполения (Ok = 0, Error = -1, etc..)
- [GuiAction] объявленное действие. Опциональное.

Если объявленное действие не является пустым оно будет выполнено незамедлительно. Таким образом мы затираем предыдущее действие новым начиная с шага (2). В противном случае выполнение завершается со статусом возвращенным из последнего выполненного действия.

Список GUI типов действий

Полный список GUI типов действий и их явных явных свойств описаны ниже:
1. PluginOpenFolderAction

Данные:

```
class PluginOpenFolderActionData
{
	String caption; // optional; default = null (auto)
	String media_url; // optional; default = null (auto)
}
```

Действие:
отображается внутри папки плагина. В больших деталях:
- заголовок папки добавляется в блок отображающий путь;
- В Get_folder_view($media_url) выполняется операция плагина;
- данные вывода используются для показа содержимого папки

Параметры:
'media_url':
- если не пустой используется, как идентификатор папки, которая открывается. В противном случае media_url в текущий момент выбранного в папке элемента является использованным. Если нет выбранных элементов, тогда действие игнорируется.

'caption':
- используется как путь элемента если не пустое. В противном случае применяется заголовок текущего выбранного в папке элемента.

Возвращаемые данные:
действие отправки: никогда
статус: ok/failed

2. PluginTvPlayAction

Данные:

```
class PluginTvPlayActionData
{
   String initial_group_id; // optional, default = null (auto)
   String initial_channel_id; // optional, default = null (auto)
   Boolean initial_is_favorite; // optional, default = false;
   unixtime_t initial_archive_tm; // optional, default = -1 (unset)
}
```

Действие:

запускает ТВ проигрывание. Состояние выбора и проигрывания инициализируется с помощью полученных параметров. Более детально:
- media_url текущего выбранного элемента папки применяется; если ничего не выбрано тогда $media_url является пустым.
- Get_tv_info($media_url) выполняется операцию плагина;
- Если данные вывода не правильные действие будет не успешным.
- состояние ТВ проигрывания инициализируется используя полученные данными PluginTvInfo.
- если некоторые инициализируемые_XXX параметры устанавливаются, тогда они заменяют выбор по-умолчанию и состояние проигрывания.

Возвращаемые данные:

действие отправки: никогда
статус: ok/failed

3. PluginVodPlayAction

Данные:

отсутствуют

Действие:

запускает VOD проигрывания. Более детально:
- media_url текущего выбранного элемента папки применяется; если ничего не выбрано тогда $media_url является пустым.
- Get_vod_info($media_url) выполняется в этой функция плагина.
- если данные вывода не правильные, тогда действие не выполняется.
- статус VOD проигрывания инициализируется используя полученные данные PluginVodInfo.

Возвращаемые данные:

действие отправки: никогда
статус: ok/failed

4. PluginHandleUserInputAction

Данные:

отсутствуют

Действие:

запускает Handle_user_input(GuiAction::$params) функцию плагина. Возвращаются данные функции в формате действия отправки.

Возвращаемые данные:

действие отправки: полученное из функции Handle_user_input()
статус: ok/failed

5. PluginShowErrorAction

Данные:

```
class PluginShowErrorActionData
{
    Boolean fatal;
    String title;
    List<String> msg_lines; [optional]
}
```

Действие:

показывает ошибку пользователю. В режиме проигрывания это одно строчная текстовая надпись отображаемая по центру экрана; в режиме Dune меню диалоги отображаются с полученными заголовком и строками сообщения.
- если (fatal == true) тогда происходит выход из плагина посредством обработки ошибки планировщиком. В этом случае проигрывание прекращается и происходит перемещение в верхний каталог Dune.
- возвращает статус ошибки.

Возвращаемые данные:

действие отправки: никогда
статус: ok/faile

6. ShowDialogAction

Данные:

```
class ShowDialogActionData
{
    String title;
    List<GuiControlDef> defs;
    Boolean close_by_return; // optional, default = false
    int preferred_width; // optional, default = 0 (auto)
}
```

Действие:

отображает специфические элементы диалога и взаимодействует с пользователем до момента закрытия.

Возвращаемые данные:

действие отправки: если диалог закрыт в зананчении CloseDialogAndRun GUI, тогда возвращается

CloseDialogAndRunActionData::post_action в действии отправки.
статус: ok

7. CloseDialogAndRunAction

Данные:

```
class CloseDialogAndRunActionData
{
    GuiAction post_action; // optional, default = null (none
    )
}
```

Действие:

Закрывает текущий диалог и возвращает 'post_action', как результат вызова SHOW_DIALOG GUI действия .
Это действие игнорируется, когда выполнение извне элементов диалога или папки интерактивных элементов.

Возвращаемые данные:

действие отправки: нет.
статус: ok

8. ResetControlsAction

Данные:

```
class ResetControlsActionData
{
    List<GuiControlDef> defs;
    int initial_sel_ndx; // optional, default = -1 (do not
    change selection)
    GuiAction post_action; // optional, default = null
    (none)
}
```

Действие:

заменяет статус текущего элемента диалога или папки интерактивных элементов.

Это действие игнорируется, когда выполнение извне элементов диалога или папки интерактивных элементов.

Возвращаемые данные:

действие отправки: из действия данных
статус: ok

9. ShowPopupMenuAction

Данные:

```
class ShowPopupMenuActionData
{
    List<GuiMenuItemDef> menu_items;
    int selected_menu_item_index; // optional, default
    = 0
}
```

Действие:

отображает всплывающее меню и взаимодействует с ним до момента пока пользователь не закроет его. Если пользователь выбирает пункт меню, тогда происходит отправка действия в возврате результата, как действие отправки.

Возвращаемые данные:

действие отправки: действие назначенное пункту меню выбранного пользователем.
статус: ok

10. StatusAction

Данные:

```
class StatusActionData
{
	int status;
}
```

Действие:

только возвращает полученный статус

Возвращаемые данные:

действие отправки: нет.
статус: ok

11. PluginUpdateFolderAction

Данные:

```
class PluginUpdateFolderAction
{
    PluginRegularFolderRange range;
    Boolean need_refresh;
    int sel_ndx; // optional, default = -1 (do not
    change current)
}
```

Действие:

Частично заменяет состояние текущей регулярной папки плагина:
- если need_refresh установлено в TRUE, хешированное состояние папки очищается.
- полученный диапазон является задействован или заменен.
- если sel_ndx >= 0 тогда фокус перемещается на элемент для которого получен индекс

Это действие игнорируется, когда выполнение извне регулярной папки.

Возвращаемые данные:

действие отправки: нет.
статус: ok

12. PluginInvalidateFolders

Данные:

```
class PluginInvalidateFoldersActionData
{
	List<String> media_urls;
	GuiAction post_action; // optional, default = null (none)
}
```

Действие:

Очищает данные кеша, которые плагин идентифицировал через получение media_urls, производя полную перегрузку данных.
Это действие нужно для случая, когда произошли некоторые действия внутри папки потомка у которого произошла замена статуса родительской папки. Использовать это действие возможно для принуждения родительской папки к перезагрузке этого состояния.

Возвращаемые данные:

действие отправки: из действия данных.
статус: ok

13. PluginRunNativeCode GUI action.

Данные:

нет

Действие:

загрузка библиотеки 'native_code.so' из папки где установлен плагин и пытается выполнить некоторые функции из этой библиотеки.

Возвращаемые данные:

действие отправки: никогда
статус: ok/faile

14. PluginClearArchiveCache GUI action.

Данные:

```
class PluginClearArchiveCacheActionData
{
	String archive_id; // optional, default = null (all)
	GuiAction post_action; // optional, default = null (none)
}
```

Действие:

Удаляет кешированные данные архивов плагина из файловой системы (если существуют) . Если 'archive_id' установленное значение только указанные архивы очищаются.

Возвращаемые данные:

действие отправки: из действия данных.
статус: ok

GUI элементы управления

GUI элементы управления могут появляется в следующих GUI контейнерах:
- папка элементов управления
- диалоги элементов управления

Вид и поведение элементов управления папок и диалогов очень похожи. GUI элементы управления размещаются с верху до низу имея 1 или 2 колонки и каждая строка является выровненная влево. Каждый элемент может иметь опцию «Подпись», текстовую строку. Когда подпись не указанна GUI элемент располагается в 1-й колонке и ничего не добавляется в 2-й. Длина GUI элемента 2-колонки может быть автоматически упорядочена(если GUI элементу не установлена длинна явно).

Папка элементов управления открывается с помощью PLUGIN_OPEN_FOLDER GUI действия. Последовательный вызов Get_folder_view() функции возвращающей среди количества данных другие данные, указывающие папку элементов управления.

```
class PluginControlsFolderView
{
        Array<GuiControlDef>    defs;
        int                     initial_sel_ndx;
}
```

Папка элементов управления может быть закрытой в такой же способ, как и другие папки Dune: с помощью RETURN, TOP_MENU.

Диалог элементов управления открывается с помощью SHOW_DIALOG GUI действия:

```
    class ShowDialogActionData
    {
        String title;
        List<GuiControlDef> defs;
        Boolean close_by_return; // optional, default = false
        int preferred_width; // optional, default = 0 (auto)
    }
```

Если 'close_by_return' установлено true тогда диалог может быть закрыт через RETURN; в других случаях это не возможно. Единственный другой способ закрыть диалог вызвать CLOSE_DIALOG_AND_RUN GUI действие. Обычно действия этого типа назначаются по крайней мере одной из кнопок диалога.

Ядро папки элементов управления и диалогов конкретизирована в список GUI элементов управления определенного по типу:

```
    class GuiControlDef
    {
        String name;
        String title;
        GuiControlKind kind;
```

```
    Object specific_def;
}
```

Поле 'name' определяет идентификатор GUI элемента. Оно применяется для прохождения значения GUI элемента управления к GUI действиям вызванных внутри контейнера.

Поле 'title' определяет дополнительную строку подписи.
Поле kind' определяет вид GUI действия:

```
enum GuiControlKind
{
    GUI_CONTROL_LABEL,
    GUI_CONTROL_COMBOBOX,
    GUI_CONTROL_TEXT_FIELD,
    GUI_CONTROL_BUTTON,
    GUI_CONTROL_VGAP
}
```

Поле 'specific_def' содержит дополнительные параметры указывающие при отдельного случае GUI элементов управления вида.

Метка

Данные:

```
class GuiLabelDef
{
    String caption;
}
```

Детали:

GUI метка отображается как фиксированная простая текстовая подпись. Если строка является так длинной, она обрезается в центре.

Параметры:

Выпадающий список

Данные:

```
class GuiComboboxDef
{
          String initial_value; // Optional, default =
  null (first)
    List<Pair<String, String>> value_caption_pairs;
    int width; // optional; default = -1 (auto)
          GuiAction confirm_action; // optional; default =
  null (none)
          GuiAction apply_action; // optional; default =
  null (none)
}
```

Детали:

GUI выпадающий список отображается, как стандартный Dune выпадающий список. Он может быть в 3 состояниях:

-без фокуса: основной компонент отображаются без фокуса.
- с фокусом: основной компонент отображаются с фокусом. Кнопка ENTER перехватывается при переключениях для сохранения состояния.
- архив: основной компонент отображается с фокусом и всплывающим меню. Кнопки ENTER и RETURN перехватываются.

Параметры:

Поле 'value_caption_pairs' - модель выпадающего списка: список пар *{выбор_значение, выбор_заголовок}*. *выбор_заголовок* отображается на экране и *выбор_значение* применяется, как идентификатор выбора.

'initial_value' - в*ыбор_значение,* которое будет выбрано по-умолчанию.

'width' - длинна выпадающего списка, если значение не указано, применяется значение по-умолчанию. На самом деле длинна выпадающего списка не может быть большей меньшей, чем значение по-умолчанию.

'confirm_action' - если указано значение по-умолчанию, применяется, когда пользователь нажимает ENTER в выпадающем меню. Через выполнения подтверждение действия не производится и выпадающий список сохраняет статус в «активное состояние». Если статус выполнения OK, тогда выбор применяется (выпадающий список собирает внутри на выбранное состояние и основной компонент обновляется); В противном случае ничего не применяется и выпадающий список переходит в запомненное в архиве состояние.

'apply_action' - указывает, что должно быть выполнено после применения изменений. Статус выполнения игнорируется.

Текстовое поле

Данные:

```
class GuiTextFieldDef
{
      String initial_value; // Optional, default = null
      (first)
      Boolean numeric; // Optional, default = false
      Boolean password; // Optional, default = false
      Boolean has_osk; // Optional, default = false
      Boolean always_active; // Optional, default =
      false
      int width; // optional; default = -1 (auto)
      GuiAction confirm_action; // optional; default =
      null (none)
      GuiAction apply_action; // optional; default =
      null (none)
}
```

Детали

GUI текстовое поле отображается, как стандартное Dune GUI текстовое поле. Оно может быть в 3 состояниях:

-без фокуса: основной компонент отображаются без фокуса.
- с фокусом: основной компонент отображаются с фокусом. Кнопка ENTER перехватывается при переключениях для сохранения состояния.
- архив: основной компонент отображается с фокусом и всплывающим меню. Кнопки ENTER и RETURN перехватываются.

Параметры:

'initial_value' — строка инициализации.
numeric' — если true, тогда только числовые значения могут быть добавленные.
pasword' — если true, тогда символы отображаются как '*';
'has_osk' — если true, тогда на экрана отображается клавиатура под текстовым полем. Отметьте: на данный момент OSK высота не принимается во внимание, когда компоненты проектируются. Так иногда нужно добавлять VGapis для получения нормального отображения контейнера.

'always_active' — если true, тогда «фокус» текстового поля не отображается, например текстовое поле автоматически отображается активным, когда получает фокус.
'width' — указывает длинну текстового поля, если поле не указано применяется некоторое предопределенное значение.

'confirm_action' — если указано, используется когда пользователь нажимает ENTER в активном состоянии элемента. Благодаря подтверждению действия выполненные изменения в текстовых не фиксируются и текстовое поле запоминает их в активное состояние. Если выполнение успешно, тогда применяются изменения и текстовое поле получает фокус; в противном случае изменения возвращаются и текстовое поле возвращает предыдущее значение.

'apply_action' — если указано, выполняется после применения изменений. Статус выполнения действия игнорируется.

Кнопка

Данные:

```
class GuiButtonDef
{
    String caption;
    int width; // optional; default = -1 (auto)
    GuiAction push_action; // optional; default = null (none)
}
```

Детали:

GUI кнопка отображается, как стандартная Dune кнопка. Она может быть в таких состояниях:
- без фокуса: кнопка отображается без фокуса.
- с фокусом: кнопка отображается с фокусом; ENTER кнопка перехватывается для выполнения push_action.

Параметры:

'capiton' — текстовая подпись на кноке.
'width' — длинная кнопки; если не указана, тогда применяется длинна по-умолчанию.
'push_action' — если указана, выполняется когда пользователь нажимает ENTER на сфокусированной кнопке.

VGap

Данные:

```
class GuiVGapDef
{
    int vgap;
}
```

Детали:

Этот псевдоэлемент применяется для изменения текущего расстояния между GUI элементами управления. Он не отображается, как все остальные.

Параметры:

'vgap' — числовое значение, которое добавляется, к расстоянию по-умолчанию между ближайшими элементами. Если число больше ноля, тогда расстояние увеличивается, если негативное число, тогда расстояние уменьшается.

GUI действия выполняемые в контейнерах GUI элементов

Каждое действие выполненное из GUI элемента берет дополнительные параметры в 'params', ключ-значение, списку.

Значение 'selected_control_id' назначается для GuiControlDef::name, для текущего выбранного элемента. Этот элемент всегда производит это действие.

Для каждого GUI элемента по типу (выпадающий список или текстовое поле) значение 'GuiControlDef::name' устанавливается для текущего статусного значения этого элемента. Статусное значение текстового поля является текущим статусом элемента. Статусное значение для выпадающего списка является текущее выбранное значение.

Фактически только GUI действие, которое используется в контейнере GUI элемента является HANDLE_USER_INPUT. Когда плагин создает HANDLE_USER_INPUT действие, оно будет назначено к некоторому GUI элементу, который должен предварительно добавить некоторые маркеры в 'params' в порядке, доступные в будущем благодаря узнанным контекстом действия вызываемые из реализации функции Handle_user_input(). Таким образом, реализация Handle_user_input() может отличаться для разных для разных GUI элементов управления контейнера и иметь всю информацию о текущем состонии GUI элемента в текущем контенере.

Разнообразные особенности

Предустановленные ресурсы плагина

- Каждый из плагинов, может содержать некоторые ресурсы (картинки). Эти картинки могут быть полученные через специальный url синтаксис.

plugin_file://path/to/resource

или

plugin_file://%plugin_name%/path/to/resource

Первый синтаксис доступа к ресурсам для текущего плагина (контекстная-зависимость).

Второй синтаксис открывает доступ к предустановленным ресурсам плагина со специфическим именем.

Архивы плагинов

На данный момент это вариант кеширования картинок использованных для представления меню папок плагина в системном хранилище или флеш памяти (если поддерживается). PluginFolderView и PluginTvInfo могут иметь 'archive' поле PluginArchiveDef типа:

```
{
    string $id;
    map(string,string) $urls_with_keys;
    string $all_tgz_url; // optional
    long long total_size;
    # etc..
}
```

Если один не применяет архив плагина для папки, это «живет» в PluginFolderView::archive, как null и пишет http://xxx/yyy.png url картинки для отправки ViewItemParams::icon_path. В этом случае картинка будет http загруженной много раз.

Использовать архив плагина, возможно если указать PluginFolderView::archive:

```
array(
      'id' => 'myarchive',
      'urls_with_keys' =>
        array(
          'xxx_key.png' => 'http://xxx/yyy.png',
          # etc
        ),
      'all_tgz_url' => 'http://xxx/all.tgz', // optional
      'total_size' => '1234567',
)
```

и специальный синтаксис для ViewItemParams::icon_path:

plugin_archive://myarchive/xxx_key.png

Выполнение PLUGIN_OPEN_FOLDER/PLUGIN_TV_PLAY операции сначала проверяет указан ли архив и запускает диалог «Обновления архива» если нужно. Диалог «Обновления архива» проверяет менялся ли архив. Это проверяется если подходящее хранилище существует и имеется достаточно свободного места. Тогда выполняется синхронизация с удаленным архивом и локальным хешем(файлы высокого качества удаляются, недостающие файлы загружаются). Файловые имена используются, как ключи в локальном кеше. Значение 'total_size' указывает размер для возможно накопленного размера всех файлов в архиве.

Файлы могут быть загруженные следующими двумя разнообразными способами:

- первый, если число картинок для загрузки является большим чем некоторый лимит (сейчас 50), тогда gzip архив загружается из all_tgz_url' URL, распаковывается и применяется для нового локального хеша. Содержимое архива должно быть с ключами 'urls_with_keys', которые должны содержать такие наборы файлов.

- другим способом (если < 50 файлов должно быть загружено или 'all_tgz_url' не укзана) нужно для загрузки из удаленного хранилища указать в 'urls_with_keys'.

Обращение к URL иконок в виде 'plugin_archive://<archive>/<key>' работает корректно каждый раз когда Dune GUI было неспособным получить локальную копию архива. В этом случае plugin_archive:// URL работает, как http:// URL.

Асинхронная загрузка картинок

Асинхронная загрузка картинок используется для регулярных папок с FolderViewParams::async_icon_loading == true. Если async_icon_loading == false тогда GUI заблокирован до момента пока все видимые иконки загружаются.
Если async_icon_loading == true тогда файлы иконок загружаются в отдельном потоке. В процессе загрузки применяется прараметры PluginRegularFolderView::not_loaded_view_item_params для визуального представления элемента или следовательно другие картинки, которые могут быть задействованы взамен.
Отметьте:

некоторые ресурсы(картинки) являются недоступны для асинхронной загрузки.
1.) Стандартные ресурсы обложек Dune, например: URL gui_skin://xxx
2.) Предустановленные ресурсы плагина, например: plugin_file://xxx
3.) Ресурсы из уже хешированных данных архива плагина, например: plugin_archive://<archive_name>/path

Проигрывание ТВ архива

Каждый плагин ТВ каналов может иметь архивное проигрывание.
Для того, чтобы подключить архивное проигрывание:
- PluginTvChannel::have_archive должно быть установлено true.
- Get_tv_playback_url() + Get_tv_stream_url() должны принимать действительные unix временные значения для параметра 'archive_tm_sec' и возвращать корректные архивные URL потоков.

- archive_past_days и archive_delay_sec из PluginTvChannel должны быть установлены соответствующим образом. По-умолчанию установлен 14 дней и 31 минута соответственно.

Защищенные ТВ каналы

На данный момент только один из способов проверки защищенного кода для некоторых каналов, для этого нужно сделать следующее:
- установить PluginTvChannel::is_protected в true;
- установить PluginTvChannel::playback_url_is_stream_url в false, если не сделано еще.
- зашифровать полученный код внутри результата playback_url в реализации Get_tv_playback_url().
- гарантировано, что Get_tv_stream_url() возвращает защищенную строку, каждое время когда защитный код не правильный и актуальный URL потока в противном случае.

Автоматическое продолжение

Особенность автоматического продолжения включается благодаря конфигурации плагина:

```
<auto_resume>
    <enable>yes</enable>
    ...
</auto_resume>
```

Процедура автоматического продолжения запускается, когда стартует STB после полного выключения или спящего режима. Эта процедура пытается восстановить последнее состояние Dune GUI до момента выключения STB. На данный момент только следующие GUI состояния могут быть автоматически продолженными.

- Просмотр ТВ для определенного канал в реальном времени.
- Архивное ТВ проигрывание для определенных каналов и временных меток.
- VOD проигрывание для определенного видео.

Отметьте: позицию Dune GUI в иерархии папок не запоминает и она не может быть восстановленной.

Есть возможность добавить стандартный IP адрес и действующее время в требования для автоматического продолжения.

```
<auto_resume>
    ...
    <ip_address_required>yes</ip_address_required>
    <valid_time_required>yes</valid_time_required>
</auto_resume>
```

Также это предоставляет возможность указать пользовательское GUI действие для выполнения взамен действия автоматического продолжения по-умолчанию. Пример:

```
<auto_resume>
    <enable>yes</enable>
```

```
    <action>
        <type>plugin_handle_user_input</type>
    </action>
</auto_resume>
```

Все значения resume_state.properties будут переданы в действие автоматического продолжения как параметры.

Поддерживаемые медиаформаты

IPTV и Интернет проигрывание

1.) Требования к iptv

1. Мультикасты UDP (IPTV потоки интернет провайдеров)
   - URL формат для доступа: udp://@ip-address:port
   - Протоколы: мулькаст raw-UDP или мультикаст RTP-over-UDP
   - Медиа контейнер: TS
   - Поддерживаемые видео кодеки: H.264, MPEG2
   - Поддерживаемые аудио кодеки: AAC, AC3

2. HTTP/TS (предназначены для проигрывания интернет потоков Live-TV)
   - URL формат для доступа: http://ts://host[:port][/path]
   - Протокол: HTTP
   - Медиа контейнер: TS
   - Поддерживаемые видео кодеки: H.264, MPEG2
   - Поддерживаемые аудио кодеки: AAC, AC3

3. MMS (Microsoft Media Server) потоки.
   - URL формат для доступа: mms://host[/path]
   - Протокол: MMS

Отметим, что виды поддерживаемых потоков данного типа ограничены, поэтому требуется обязательное тестирование совместимости.

2.) Требования к проигрыванию интернет-контента

1. HTTP/MP4 (good for VOD content)
   - URL формат для доступа: http://mp4://host[:port][/path]
   - Протокол: HTTP
   - Медиа контейнер: MP4
   - Поддерживаемые видео кодеки: H.264, MPEG2
   - Поддерживаемые аудио кодеки: AAC, AC3

Аудипотоки (Интернет радио)

1. HTTP/{MP3,AAC,OGG}
   - URL формат для доступа: http://audio_stream://host[:port][/path]
   - Протокол: HTTP
   - Медиа контейнер: none (raw аудио потоки)
   - Поддерживаемые аудио кодеки: MP3, AAC, OGG

2. MMS/WMA
   - URL формат для доступа: mms://host[/path]
   - Протокол: MMS
   - Медиа контейнер: ASF
   - **Поддерживаемые аудио кодеки: WMA**