UIEPlayer 移植用 STB API 仕様　1

- 概要および標準 STB API 編 -

Revision 2.2.8

11/16/2009

UIEvolution,Inc.

1. はじめに

本資料では、ひかり TV 対応 UIEPlayer を Linux を OS とする受信機（STB）に移植するに当たり、受信機のファームウェアでどのような API セットを用意する必要があるかについて概要を述べたものです（UIEPlayer を除いた受信機の構成を、本資料では受信機レジデントと呼びます）。これらの API セットはライブラリとして提供され、UIEPlayer からリンクされることを想定しています。

本資料では、UJML Language Reference 2.1 に記載される標準的な UJML アプリケーションの実行に必要となる、標準 STB API について述べます。動画再生などの拡張機能に関しては、別途拡張 API 資料を参照してください。

関連資料:

- UIEPlayer 移植用標準 STB API 資料1（概要および標準 STB API 編・本資料）
- UIEPlayer 移植用拡張 STB API 資料2 (システム・メディアプレイヤー編)
- UIEPlayer 移植用拡張 STB API 資料3 (番組表データ編)
- UIEPlayer 移植用拡張 STB API 資料4 (視聴・録画予約データ編)
- UIEPlayer 移植用拡張 STB API 資料5 (録画データ編)
- UJML Language Reference 2.1 (http://developer.uievolution.com/docs/en/LangRef/index.html)

2. 用語について

用語は「IPTV 規定」にて使用されているものを原則として使用します。

UIEvolution および本資料独自の用語を以下に説明します:

- UIEPlayer ： UJBC ファイルを解釈・実行する GUI ミドルウェア。UIEngine とも呼ばれるが本資料では UIEPlayer に統一
- UJML, UJBC : UJML は XML 形式で記述される UIEPlayer 用言語。UJML は中間コード形式の UJBC にコンパイルされ、UIEPlayer 上で実行される。詳細は http://www.uievolution.com/support/ 参照
- 受信機レジデント(受信機システム) : UIEPlayer 担当部分を除く受信機の機能(ファームウェア/ソフトウェア)を表す
- GUI ： UIEPlayer 上で実行されるひかり TV アプリケーション。複数の UJBC ファイル、イメージファイルから構成される。これらのファイルは受信機レジデント内のストレージまたはネットワーク経由で読み込まれる。
- レジデント UJML： 受信機のファイルシステムに保存される UJML(UJBC)コード
- メディアプレイヤー ： 受信機レジデント中のコンテンツ処理機能（主に、映像・音声複合処理及び出力）を表す
- STB API： 本資料で述べる、UIEPlayer – 受信機レジデント間の API を表す
- Native component : UJML 言語のコンポーネントの実装を、ネイティブコードで記述したもの。UJML からネイティブコード(拡張 STB API)を呼び出すために必要となる。

3. UIEPlayer 版ひかり TV の機能

3.1 概要

「IPTVFJ　STD-0001　概説」を元に、UIEPlayer 版ひかり TV の機能を説明すると、以下のようになります:

- 配信サービスモデル (IPTVFJ STD-0001「第 2 章　想定する配信サービスモデル」)
  VOD サービス・IP 放送サービス
- 想定するネットワーク環境（IPTVFJ STD-0001「3.1　想定するネットワーク環境」）
  配信ネットワークは CDN スコープとなる。

また、IPTVFJ STD-001「4.2 IPTV 要素技術の概要」 において、UIEPlayer および受信機レジデントが担当する IPTV 要素技術は以下の通りです:

・IPTVFJ STD-0001「4.2.1 コンテンツ符号化」
受信機レジデント側で行う必要がある。

・IPTVFJ STD-0001「4.2.2 コンテンツ伝送方式」
受信機レジデント側で行う必要がある。

・IPTVFJ STD-0001「4.2.3 CAS/DRM」
原則として受信機レジデント側で行う必要がある。ただし、コンテンツ購入・解約時の MC ライセンスの再読み込み、および MC ライセンス更新情報通知サーバーからの通知に伴う MC ライセンスの再読み込みは、UIEPlayer 側が受信機レジデントに対してリクエストを行う。

・IPTVFJ STD-0001「4.2.4 システム制御情報」
VOD サービスにおける再生制御情報の取得、およびそれ以降の処理は受信機レジデント側が行う。また、ストリーム受信における特殊再生などの再生制御指示に関係する部分は、原則として受信機レジデント側が主体となる。ただし、コンテンツ再生中のダイアログなどの GUI パーツの表示は原則として UIEPlayer 側が行う。

・IPTVFJ STD-0001「4.2.5 ﾚｼﾞﾃﾞﾝﾄﾍﾞｰｽｺﾝﾃﾝﾂﾅﾋﾞｹﾞｰｼｮﾝ／4.2.6 ﾌﾞﾗｳｻﾞﾍﾞｰｽｺﾝﾃﾝﾂﾅﾋﾞｹﾞｰｼｮﾝ」
UIEPlayer 版ひかり TV では、上記のどちらでもない独自のコンテンツナビゲーションを行う。画面や個々のサービスを起動するインタフェースの呼び出しなどが記述された UJBC ファイルをサーバーもしくは受信機のファイルシステムから随時取得し、UIEPlayer がこれを解釈・実行してコンテンツナビゲーションの画面を提供する。VOD における ECG も UJBC ファイルレベルで記述されているが、サーバーサイドで ECG メタデータを処理し、UJBC ファイルに変換したものを受信機サイドで取得している。
UJBC ファイルは機種非依存となるため、UIEPlayer を備えた受信機で共通に利用が可能。

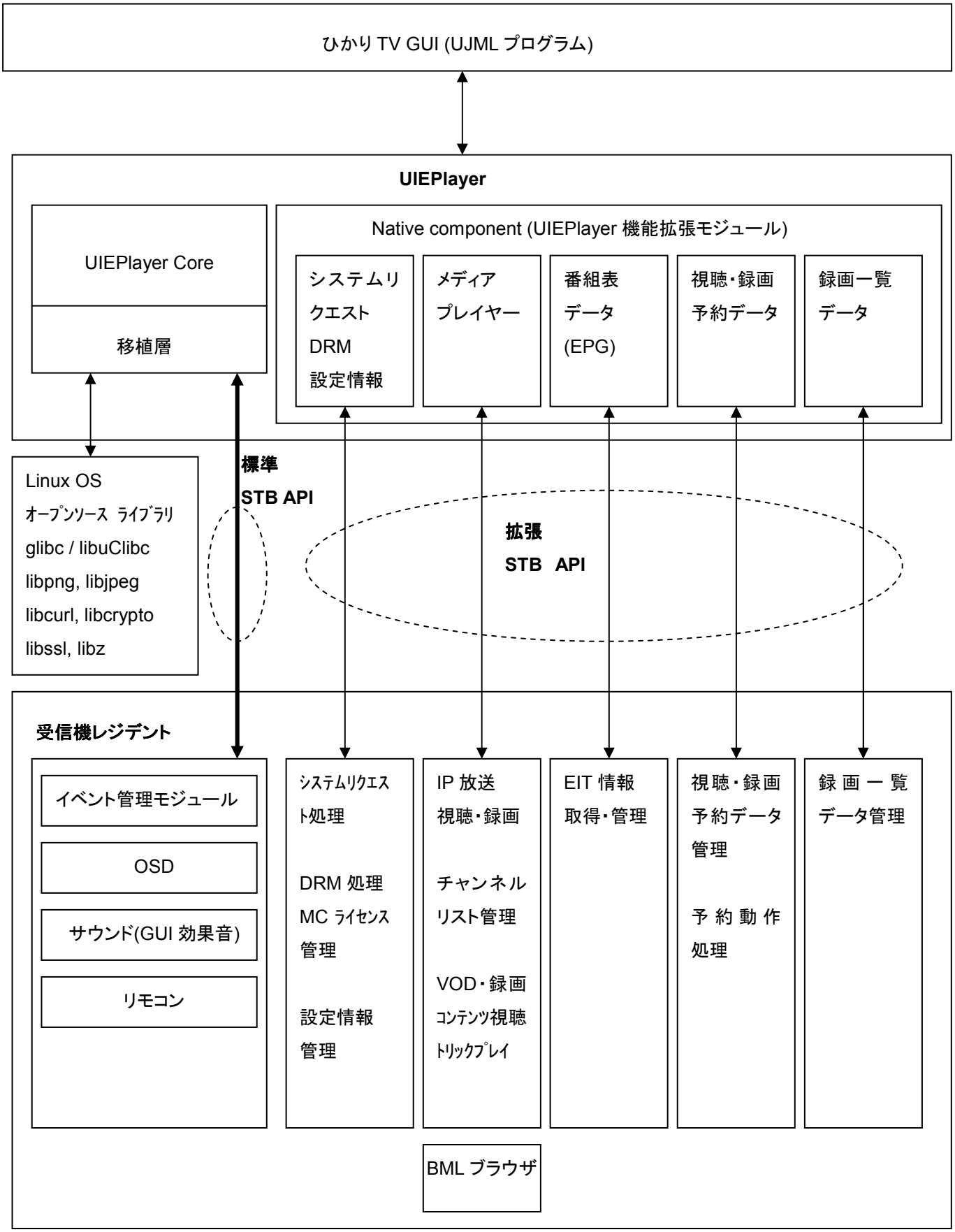
・IPTVFJ STD-0001「4.2.7 サービス発見」
受信機レジデント側が、CDN 構成情報サーバへのアクセス、および PF 構成情報サーバーへのアクセスを行う。ただし、UIEPlayer(GUI)側でもそれらの情報が必要なため、受信機レジデント内に保存された当該ファイルをアクセスすることを想定している。

・IPTVFJ STD-0001「4.2.8 ネットワーク接続」

受信機レジデント側で行う必要がある。

4. UIEPlayer と受信機レジデントの構成

UIEPlayer の視点から見た受信機のシステム構成図を以下に示します。UIEPlayer はライブラリとして受信機レジデントに組み込まれ、専用のスレッドまたはプロセスとして受信機レジデントから起動されます。受信機全体の動作は受信機レジデント主体で行われます。
本資料では、太線で記載された標準 STB API の仕様について記述されています。
UIEPlayer から受信機レジデントへのリクエストは、STB API 関数コールで行われます。一方、受信機レジデントから UIEPlayer へのリクエストやステータス通知は STB API イベントベースで行われ、UIEPlayer 側で非同期に処理されます。



5. UIEPlayer の移植に必要なリソース

UIEPlayer 側の視点からの、ひかり TV 対応受信機に必要なシステムリソースの一例を以下に示します。この値は、2009 年 6 月時点で実際にサービス中の現行の受信機での実績をベースとしたもので、受信機のアーキテクチャ・ファームウェア・ひかり TV GUI アプリケーションによって必要な値は変化します。特に、フレームバッファが 32bit / pixel の場合、CPU・メモリ共に必要リソースが増加します。

CPU:　MIPS based 300MHz (lowest)
Memory: 128MB (44MB for Linux)
Frame buffer: 16 bit per pixel (R5G6B6)

Program binary size:

- UIEPlayer:　2.5MB (Including font data and CAS/DRM static library)
  STB resident library described in this document is not included.
- GUI boot loader:　150KB
- Regident UJML: 3MB (TBD) –　受信機ファイルシステムに保存する UJBC および画像ファイル

・使用しているライブラリのリストおよびその容量

- libcrypto:　1.6MB
- libssl:　270KB
- libstdc++:　600KB
- libuClibc:　410KB
- libz:　58KB
-libjpeg　N/A
-libpng　N/A
-libCurl　N/A

Memory allocated for UIEPlayer

- Text code:　2MB
- Heap data:　16MB for 16bit color graphics (Assumed 4MB for back buffer)
  24MB for 32bit color graphics (Assumed 8MB for back buffer)
  * Memory used by the related library above is not included
- Temporary file size
  300KB
  * File created by STB resident is not included

Persistent file size (NVRAM size):

200KB　* File created by STB resident (which included reservation data) is not included

5. 標準 STB API リファレンス

5.1. 受信機イベント管理 API

UIEPlayer はイベントドリブンな動作を行います。受信機レジデント側は UIEPlayer に対してのイベントを内部に保持します。UIEPlayer は本 API を使用してイベントの取得を行い、受信機レジデント側とは非同期に動作します。なお、別途資料に記載する拡張 STB API のイベントも、すべてこの API 経由で取得されます。

イベントのデータ構造は以下の通りです。

```
typedef struct UIE_STB_EVENT_S
{
        int id,             // unique event id
        int type,           // event type
        int param,          // additional parameter for the event
        int param2,
        int param3,
        int param4,
        void *eventData;    // optional data for the event
} UIE_STB_EVENT, *UIE_PSTB_EVENT;
```

受信機イベント管理 API 関数一覧

API Name	Synopsys	Parameters	Note
uie_getEventFileDescriptors()	イベント通知準備	pollfd 配列へのポインタ（Output），配列のサイズ	ファイルディスクリプタとイベントマスクのリストを取得
uie_getEvent()	イベントの取得	- イベントへのポインタ (Output)	UIEPlayer の用のイベントキューの先頭のイベントを取得
uie_removeEvent()	イベントの削除	イベントへのポインタ	UIEPlayer の用のイベントキューの先頭のイベントを削除

int uie_getEventFileDescriptors(struct pollfd *pfds, int num)

Copies those pollfds (file descriptor and events) in the array of pollfd structure, with which UIEPlayer needs to wait for

events for maximum performance. The array of pollfd will be specified when UIEPlayer calls poll(). UIEPlayer will call uie_getEvent() to retrieve an event. If the function returns zero (no fd for STB event), the UIEPlayer will periodically poll STB event by calling uie_getEvent() described later

UIEPlayer が受け取るべきイベントの発生に際して、そのイベントを poll() システムコールで待つことができるように、ファイルディスクリプタ とイベントマスクのリストを取得します。このドキュメントに記載しているイベントのほかにも、UIEPlayer が独自に poll() したいものがあるので、パフォーマンス追求のためにそれらと一緒にして待ちたいというのが目的です。
なお、STB 側で用意するイベントモデルが poll()でのポーリングに対応できない場合、本 API では 0 を返すようにしてください。その際は、次に説明する uie_getEvent()を UIEPlayer 側で定期的にポーリングするようになります。

parameters:
[OUTPUT] struct pollfd *pfds: pointer to an array of pollfd
[INPUT] num : the size of the array pfds[] (number of elements)
Returns:
zero or positive value less than or equal to num :
number of file descriptors copied to the array pfds[]
zero or positive value bigger than num :
total number of file descriptors (UIEPlayer will call this function again after increasing the array size)
negative value: indicates error

int uie_getEvent(UIE_PSTB_EVENT pEvent)

Returns the first item in the event queue for UIEPlayer. If there is no event item for UIEPlayer, the function will return immediately. Note this function will not remove the event from the queue – uie_removeEvent() described later will remove the event.
UIEPlayer 用のイベントキューの先頭のイベントを取得します。イベントがない場合は即座に帰ります。このファンクションでは、イベントの消去は行いません。(後述の uie_removeEvent() にて、関連するリソースなどの開放を行うため)。イベントの種類に関しては、これ以降に説明する各 API のドキュメント中に記載します。

parameters:
[OUTPUT] PSTB_EVENT pEvent:
Pointer to STB_EVENT structure allocated by the caller. All fields in the structure will be over-written by the function.

Returns:
positive value: If successful. The value is number of events (including this one) in the queue.
zero : indicates there is no event for UIEPlayer
negative value if an error happens

int uie_removeEvent(UIE_PSTB_EVENT pEvent)

Removes the specified event from the event queue. The system may also free up the memory pointed by pEvent and other system resource associated with this event. The event specified is always the event at the top of the queue.

parameters

[INPUT] PSTB_EVENT pEvent : Pointer to an STB_EVENT. pEvent always points at the top of the event queue.

Returns:

zero or positive value - indicates the function is successful and the number of events left in the queue

negative value if an error happens

event functions calling sample

The sample code below is shown here only to indicate how to call each function. It is not the actual code of UIEPlayer.
このサンプルコードは、各ファンクションの呼び方を示すために掲げているもので、実際の UIEPlayer のコードではありません。

```
#include <sys/poll.h>

typedef struct _STB_EVENT
{
int id;                 // unique event id
int type;               // event type
int param;              // additional parameter for the event
void *eventData; // optional data for the event
} STB_EVENT, *PSTB_EVENT;

int uie_getEventFileDescriptors( struct pollfd *pfds, int num);
int uie_getEvent( PSTB_EVENT pEvent );
int uie_removeEvent( PSTB_EVENT pEvent );

#include <assert.h>
#include <fcntl.h>
#include <sys/poll.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

#define FD_COUNT_INIT 5

int main()
{
    struct pollfd *pfds;
    int     numfds;

    pfds = (struct pollfd *)malloc( sizeof(struct pollfd) * FD_COUNT_INIT);
    if ( pfds != NULL )
    {
        int result;
        int num;

        num = FD_COUNT_INIT;
        result = uie_getEventFileDescriptors( pfds, num);
        if ( result < 0 )
        {
            printf( " uie_getEventFileDescriptor returns error=%d¥n", result );
            free( pfds );
            exit(0);
        }
        if ( result > FD_COUNT_INIT )
        {
            free( pfds );
            pfds = (struct pollfd *)malloc( sizeof(struct pollfd) * result );
            num = result;
            result = uie_getEventFileDescriptors( pfds, num );
            if ( result < 0 )
            {
                printf( " uie_getEventFileDescriptor returns error=%d¥n", result );
                free( pfds );
                exit(0);
            }
        }
    numfds = result;
    }

    int poll_result = 0;
    for(;;) {

        poll_result = poll( pfds, numfds, 1000);
        if ( poll_result > 0 )
        {
            int left;
            STB_EVENT StbEvent;
            while ( (left = uie_getEvent( &StbEvent )) > 0 )
```

```
            {
                printf("Got event id:%d¥n", StbEvent.id);break;
            }
        }
    }
}
```

5.3. Sound Player

Functions in this category specify how to output PCM sound data. This function is for sound effect of GUI.
このセクションで説明する関数は、GUI の効果音再生で使用されます。

Sound Player API 関数一覧

API Name	Synopsys	Parameters	Note
uie_Sound_open()	音声デバイスの初期化とオープン	なし	
uie_Sound_start()	音声再生	音声ファイルへのポインタ, サイズ	
uie_Sound_stop()	音声再生停止	なし	
uie_Sound_close()	音声デバイスのクローズ	なし	
uie_Sound_getVolume()	音声ボリューム値の取得	なし	受信機に音声変更機能がない場合、本 API[は常に 100 を返す
uie_Sound_setVolume()	音声ボリューム値の設定	設定値	受信機に音声変更機能がない場合、本 API[は常に 100 を返す

Sound Player 関連 API イベント一覧

Event Name	Synopsys	Parameters	Note
SOUND_EVENT	音声再生の開始・終了を示す	音声再生の開始・終了を示すフラグ	

int uie_Sound_open()

Opens and initializes sound device.

Parameters:

None

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_Sound_start(unsigned char *pcm_data, int size)

Starts playing the specified PCM sound data. PCM data is 44100Hz, 16bit, Stereo, Windows “WAV” data format compatible.

This function returns immediately without waiting for start or finish of the sound output. Player will receives the following events via uie_getEvent() function.

SOUND_NOTIFY_STARTED
SOUND_NOTIFY_FINISHED

If this function is called twice, the second time before the first sound has finished playing, the interrupted playback is automatically stopped before the second starts, and the corresponding events will be queued.

parameters:

[INPUT] unsigned char *pcm_data : PCM data. This data is volatile so the sound player should copy the data.

[INPUT] int size: data size in bytes.

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_Sound_stop()

Stops the current playback.

This function is non-blocking and returns immediately after the stop request is issued.

When the sound output is actually stopped, an event SOUND_NOTIFY_FINISHED will be queued for UIEPlayer.

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_Sound_close()

Closes sound device.

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_Sound_getVolume()

Gets the sound volume. The sound volume will be applied to all sound output via uie_Sound_start() call.
This is a fixed point number to two decimal places (100 represents 1.00).

Returns:
Current volume level - 0 is minimum (no sound) and 1.00 is maximum.

Note:
If the STB doesn’t have its own volume control, this API always returns 100.
受信機本体に音量変更機能がない場合、本 API は常に 100 を返す。

int uie_Sound_setVolume(int value)

Sets the sound volume. The sound volume will be applied to all sound output via uie_Sound_start() call.

Parameters:
int value : sound volume level - This is a fixed point number to two decimal places (100 represents 1.00).

Returns:
Current volume level

Note:
If the STB doesn’t have its own volume control, this API always returns 100.
受信機本体に音量変更機能がない場合、本 API は常に 100 を返す。

sound events

After calling uie_Sound_start(), player will receive the following events via uie_getEvent() function.

event type:

SOUND_EVENT 0x200

event param:

0 - SOUND_NOTIFY_STARTED: Notifies UIEPlayer the sound output starts

1 - SOUND_NOTIFY_FINISHED: Notifies UIEPlayer the sound output is finished

5.4. Remote Controller IR input

リモートコントローラーのキー入力は、まず最初に受信機レジデント側で処理すべきキーコードかどうかをチェックされます。「電源」など、常に受信機レジデント側で処理すべきキーは、UIEPlayer に渡されることはありません。
なお、受信機レジデント側独自の OSD メニューダイアログ、エラーダイアログなどが存在し、それが表示中の場合は、すべてのキーコードは受信機レジデント側で処理され、UIEPlayer には渡されないことを想定しています。

Remote Controller 関連 API イベント一覧

Event Name	Synopsys	Parameters	Note
KEY_EVENT_KEYPRESS	キー入力があったことを通知	キーコード	
KEY_EVENT_KEYDOWN	キーが押された際に発生	キーコード	必ず次の KEY_EVENT_KEYUP とペアで通知
KEY_EVENT_KEYUP	キーが離された際に発生	キーコード	

Remote Controller IR events

If Remote Controller IR inputs are handled via uie_getEvent() function, the following event code is used.
リモートコントローラーのキー入力情報は、uie_getEvent()関数によるイベントとして取得されます。使用されるイベントコードは以下の通りです。

event type:

KEY_EVENT_KEYPRESS 0x300 :

Notifies UIEPlayer the Remote controller key is pressed. It is necessary to support auto repeat is the key is being pressed..

リモートコントローラのキー入力があったことを通知します。キーが押されている間、オートリピートをサポートする必要があります。

KEY_EVENT_KEYDOWN 0x301 :

Notifies UIEPlayer the Remote controller key is just pressed.

リモートコントローラのキーが押された際に発生します。必ず、次の KEY_EVENT_KEYUP とペアで通知される必要があります。

KEY_EVENT_KEYUP 0x302 :

Notifies UIEPlayer the Remote controller key is just released.

リモートコントローラのキーが離された際に発生します。

受信機レジデント側は、KEY_EVENT_KEYDOWN/UP のペアか、KEY_EVENT_KEYPRESS 単体のどちらか一方を UIEPlayer に通知する必要があります。
KEY_EVENT_KEYPRESS イベントを使用する場合、受信機レジデント側でキーのオートリピート処理を行う必要があります。一方、KEY_EVENT_KEYDOWN/UP イベントを使用する場合、UIEPlayer 側でオートリピート処理を行います。キーリピートは原則として初回 500ms, 2 回目以降 150ms となります。
なお、後述のキーコード一覧に示す通り、オートリピートの対象となるキーは限定されています。STB レジデント側で KEY_EVENT_KEYPRESS を通知してオートリピート処理を行う場合も、これに従う必要があります。

event param:

remote controller key code listed below.

キー名称	ヘッダーファイル定義	キーコード	ｵｰﾄﾘﾋﾟｰﾄ	UJML でのキーコード	備考
1	STB_KEY_CODE_1	0x31	○	1	
2	STB_KEY_CODE_2	0x32	○	2	
3	STB_KEY_CODE_3	0x33	○	3	
4	STB_KEY_CODE_4	0x34	○	4	
5	STB_KEY_CODE_5	0x35	○	5	
6	STB_KEY_CODE_6	0x36	○	6	
7	STB_KEY_CODE_7	0x37	○	7	
8	STB_KEY_CODE_8	0x38	○	8	
9	STB_KEY_CODE_9	0x39	○	9	
0	STB_KEY_CODE_0	0x30	○	0	
11	STB_KEY_CODE_11	0x8b	○	G	
12	STB_KEY_CODE_12	0x8c	○	H	
＊	STB_KEY_CODE_STAR	0x2a	○	STAR	未使用の予定
＃	STB_KEY_CODE_POUND	0x23	○	POUND	未使用の予定
↑	STB_KEY_CODE_UP	0x01	○	UP	
↓	STB_KEY_CODE_DOWN	0x02	○	DOWN	
←	STB_KEY_CODE_LEFT	0x03	○	LEFT	
→	STB_KEY_CODE_RIGHT	0x04	○	RIGHT	
決定	STB_KEY_CODE_FIRE	0x05	○	FIRE	
戻る	STB_KEY_CODE_F1	0x06		F1	
ホーム	STB_KEY_CODE_F2	0x07		F2	PC-STB4 では「メニュー」
青	STB_KEY_CODE_GAME_A	0x10	○	GAME_A	
赤	STB_KEY_CODE_GAME_B	0x11	○	GAME_B	
緑	STB_KEY_CODE_GAME_C	0x12	○	GAME_C	
黄	STB_KEY_CODE_GAME_D	0x13	○	GAME_D	
消音	STB_KEY_CODE_MUTE	0x81			TV の機能なので定義しない
音量＋	STB_KEY_CODE_VOLUME_UP	0x84	○		TV の機能なので定義しない
音量－	STB_KEY_CODE_VOLUME_DOWN	0x85	○		TV の機能なので定義しない
SECRET	STB_KEY_CODE_SECRET	0x86		X	
メニュー	STB_KEY_CODE_MENU	0x8d		T	PC-STB4 では「詳細」
番組表	STB_KEY_CODE_EPG	0x8f		E	
録画リスト	STB_KEY_CODE_CONTENTS_LIST	0x99		L	

キー名称	ヘッダーファイル定義	キーコード	ｵｰﾄﾘﾋﾟｰﾄ	UJML でのキーコード	備考
無効キー	STB_KEY_CODE_NONE	0xff			仮想キー。UIEPlayer では無視される 未使用の予定
再生／一時停止	STB_KEY_CODE_PLAY	0x88		P	
早送り	STB_KEY_CODE_FAST_FORWARD	0x89		F	
巻き戻し	STB_KEY_CODE_REWIND	0x87		R	
停止	STB_KEY_CODE_STOP	0x8a		S	
録画	STB_KEY_CODE_REC	0x9a		Q	
ﾁｬﾌﾟﾀｰ戻し	STB_KEY_CODE_PREV_CHAPTER	0x90		J	
ﾁｬﾌﾟﾀｰ送り	STB_KEY_CODE_NEXT_CHAPTER	0x91		K	
~~先頭ﾁｬﾌﾟﾀｰ~~	~~STB_KEY_CODE_FIRST_CHAPTER~~	~~0x92~~		~~M~~	
［d］	STB_KEY_CODE_DATA	0x8e		C	
CH＋	STB_KEY_CODE_CHANNEL_UP	0x82	○	U	
CH－	STB_KEY_CODE_CHANNEL_DOWN	0x83	○	D	
字幕切替	STB_KEY_CODE_CHANGE_CAPTION	0x94		O	
音声切替	STB_KEY_CODE_CHANGE_AUDIO	0x95		A	
電源	STB_KEY_CODE_POWER	0x80		W	
サービス切り替え	STB_KEY_CODE_CHANGE_SERVICE	0x93		Y	
IPTV	STB_KEY_CODE_IPTV	0x96		I	
地デジ	STB_KEY_CODE_TB	0x97		V	
BS	STB_KEY_CODE_BS	0x98		B	
ちょっと戻る	STB_KEY_CODE_SKIP_BACKWARD	0x9b		N	
30 秒スキップ	STB_KEY_CODE_SKIP	0x9c		M	

表5－4－1　キーコード一覧

5.5. HTTP/HTTPS

UIEPlayer uses the following libraries for HTTP and HTTPS request.

curl 7.18.1

openssl 0.9.7i

なお、User-Agent は、uie_System_getProperty(“useragent”)で得られた文字列の末尾に、以下の UIEPlayer バージョン文字列を付加したものが使われます。

UIEPlayer-<UIEPlayer Version>-<UIEPlayer BuildNo>

See Also:

uie_System_getProperty(“useragent”)

5.7 Frame buffer control

UIEPlayer supports dumb frame buffer by default. Movie plane for the Media Player is placed under the frame buffer. Internally UIEPlayer supports pixel format of ARGB8888 and its graphics implementation layer converts it to actual pixel format (A1R5B5G5 for example).

If the frame buffer doesn't support alpha layer, specific transparent color code should be required (e.g. R5B6G5 = 000). Use of alternative frame buffer (i.e. back buffer) is recommended to avoid flicker.
フレームバッファがαレイヤーを備えていない場合、透明を表す特定のカラーコードが必要となります。
また、描画の際のちらつきを避けるためにバックバッファの使用が推奨されます。

Frame Buffer Control API 関数一覧

API Name	Synopsys	Parameters	Note
uie_OSD_open()	フレームバッファの初期化	バッファの高さ、バッファの横幅	
uie_Osd_close()	グラフィックサブシステムの終了	なし	
uie_OSD_getFrameBuffer()	フレームバッファのアドレス取得	バッファの先頭アドレス（Output）,1 ラインのバイト数（Output）	
uie_OSD_getFrameBufferSize()	フレームバッファのサイズ取得	バッファの高さ（Output）、バッファの横幅（Output）	
uie_OSD_flip()	バックバッファから実際のフレームバッファへの転送	なし	

int uie_OSD_open(int width, int height);

Initializes frame buffer (On Screen Display).
フレームバッファ(On Screen Display)を初期化します。

Parameters:

[INPUT] int width : width of frame buffer in pixel.

[INPUT] int height : height of frame buffer in pixel.

Frame buffer should support 640x480, 960x540 and 1280x720. 720x480 and 1920x1080 are recommended.
サイズは640x480, 960x540, 1280x720のサポートが必須。また720x480, 1920x1080 もサポートすることが推奨されます。

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_OSD_close();

Quits graphic sub system.
グラフィックサブシステムを終了します。

Parameters:

none

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_OSD_getFrameBuffer(void **buffer, unsigned int *pitch);

Returns frame buffer address. This might returns address of back buffer and actual frame buffer might exist alternatively. This function is called every time the UIEPlayer starts to start to draw a new frame. And uie_OSD_flip() is called after the frame drawing is finished.
フレームバッファのアドレスを取得します。ここで取得されるアドレスはバックバッファーで、実際のフレームバッファは別途存在する場合もあります。
この関数はUIEPlayerがフレーム単位の描画を開始する度によばれ、フレームの描画が終了するとuie_OSD_flip()が呼ばれます。
DirectFBの場合の実装を例にあげると、uie_OSD_getFrameBuffer()でIDirectFBSurface.Lock()が、uie_OSD_flip()では IDirectFBSurface.Unlock()とIDirectFBSurface.Flip()が呼ばれます。

See Also:

uie_OSD_flip()

Parameters:

[OUTPUT] void **buffer:

Pointer to buffer address.
バッファの先頭アドレスを返します。

[OUTPUT] unsigned int *pitch:

Pointer to byte size per line to be set.

Address of n-th line is calculated like ((unsigned char *)buffer + pitch * n).

1ラインのバイト数を返します。

nライン目の先頭アドレスは (unsigned char *)buffer + pitch * n

のようにして求める必要があります。

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_OSD_getFrameBufferSize(int *width, int *height);

Returns the width and height of device screen.
フレームバッファのサイズを返します。

Parameters:

[OUTPUT] int *width – Pointer to screen size in pixel.

[OUTPUT] int *height – Pointer to screen height in pixel.

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

int uie_OSD_flip();

Requests STB resident to reflect contents of frame buffer to actual screen. If the UIEPlayer uses alternative frame buffer (i.e. back buffer), this API waits V-Sync then transfer the contents of back buffer to actual frame buffer.

本APIはUIEPlayerがフレーム単位の描画を終了した場合によばれ、uie_OSD_getFrameBuffer() が返すバッファに書き込んだ内容を実際のディスプレイ上の表示に反映させる処理を行います。UIEPlayerがバックバッファーに描画を行っている場合、本APIではV-Syncを待った後、バックバッファーから実際のフレームバッファへの転送を行うことになります。

Parameters:

none

Returns:

Returns: UIE_ERR_OK if success. UIE_ERR_NG if error.

5.10. temporary file access

受信機のファイルシステム内のあらかじめ定められたディレクトリに対し、UIEPlayer は標準ライブラリを使用して、ファイルの作成・読み込み・書き込みのアクセスを行います。
なお、拡張 STB API がサポートされている場合、uie_System_getProperty(“tmppath”)で得られたディレクトリが対象となります。

作成されるファイルは、uie_ というプリフィクス、.upb という拡張子を持ちます。UJML 言語 のプロパティバッグ機能（小規模なデータのセーブ・リストア機構）に対応するものです。ファイル名の例は次のようになります。

uie_stbgui.upb
uie_stbsys.upb

個々のファイルは、おおむね 2,3KB 程度以内のサイズで、ファイルのサイズ・内容は UJML アプリケーションの動作に伴って随時変更されます。UJML アプリケーション開発ベンダーの数が増えた場合、ファイルが追加されることがあります。

当該ディレクトリが存在し、read/write 可能であることが必要です。
また当該領域が RAM ディスクにマウントされていて、STB の再起動に伴って受信機レジデントにより初期化されることを暗黙的に前提としています。

最大サイズは 200KB を想定しています。

See Also

uie_System_getProperty()

5.12. Font

Windows UCS TTF file with the following glyph is required for UIEPlayer. UIEPlayer has original format of font data which is converted from the Windows TTF file. For HikariTV service there is licensed font already.
以下の文字集合を備えた UCS 対応 Windows TTF 形式の非プロポーショナルなフォントファイルが必要となります。この Windows TTF 形式のフォントファイルからフォントデータをあらかじめ抽出し、UIEPlayer プログラム内部にフォントデータを格納します。ひかり TV サービスに関しては、すでにライセンス済みのフォントデータが存在します。

ARIB STD-B24 第一編 第 2 部「モノメディア符号化」の表 7-4(1)漢字系集合(1)～(8)漢字系集合(8)
ARIB STD-B24 第一編 第 2 部「モノメディア符号化」の表 7-6 英数集合
ARIB STD-B24 第一編 第 2 部「モノメディア符号化」の表 7-7 JIS X0201 片仮名集合

5.16. UIEPlayer main entry

本 API 関数は、UIEPlayer 側で用意されます。受信機レジデント側がこの関数を呼び出すことで、UIEPlayer の実行が開始されます。.

int uie_playermain(int argc, char *argv[])

Parameters:

[INPUT] int argc

Number of argument.

1: Default startup URL (file:///boot.uiepk) is used which is “./boot.uiepk”

2: Starup URL is specified by argv[1]

[INPUT] char *argv[]

argv[0] : Unused

argv[1]: UIEPlayer application startup URL if argc == 2

Example: http://example.com/main.ujbc

file:////boot/main.ujbc - “/boot/main.ujbc” is used

file:///boot/main.ujbc - “./boot/main.ujbc” is used

Returns:

0 : Exitted successfully.

Non zero : Error. Required to reboot by process/thread level.

Note:

この UIEPlayer メインエントリーが呼び出し側に戻ってくるのは以下のケースになります:

正常終了:

- 受信機レジデントからの SYSTEM_EVENT_EXIT_REQUEST, SYSTEM_EVENT_KILL_REQUEST が受け付けられた場合
- UJML プログラム側から UIEPlayer に対して終了要求があった場合。この場合(受信機レジデント側から要求されることなく、自発的に終了します)。この場合、受信機レジデントは再度 uie_playermain()を呼び出す必要があります (プロセス／スレッドレベルでの再起動を推奨)。

エラー終了:UIEPlayer が属するプロセス/スレッドレベルでの再起動が必要

- メモリ確保に失敗した場合

GUI related feature:

受信機の電源がリモコンなどのユーザー操作によって OFF にされた場合、実際には画面表示が OFF になるだけで、CPU（UIEPlayer プロセス）は通常通り動作していることを想定しています。

URL は RFC1738 に準拠していますが、file URL(”file://<hostname>/<path>”)において、ホスト名の指定には対応しておらず、常に省略となります(”file:///<path>)。

07/29/2009 Revision 2.2.1

IPTV, BS キーのヘッダーファイル文字列定義の誤記を修正

イベントのデータ構造に、param2, param3, param4 を追加

07/28/2009 Revision 2.2

・標準の UJML に対応した UIEPlayer 移植のための API を分冊化

08/12/2009 Revision 2.2.3

リターンコード UIE_OK, UIE_ERROR を他の API と合わせて、UIE_ERR_OK, UIE_ERR_NG に変更。

イベント構造体名称を変更

リモコンキーに、「ちょっと戻る」「30 秒スキップ」を追加

09/14/2009 Revision 2.2.4

実際のデバイススキンに合わせてキーコードを修正

10/02/2009 Revision 2.2.5

変更なし。

10/16/2009　Revision 2.2.6

・各種関数一覧表、イベント一覧表を追加

・UIEPlayer の移植に必要なリソースに関する説明を追記。

uie_playermain()の説明で、すでに廃止となった uie_System_request(SYSTEM_REQUEST_KILL)の記述が残っていたので修正。

10/30/2009　Revision 2.2.7

変更なし。

11/16/2009　Revision 2.2.8

・地上波デジタル IP 再送信対応の際は IPv4 対応が必須となる、という旨の文言を削除。