OSAKA NDS Embedded Linux Cross Forum #4

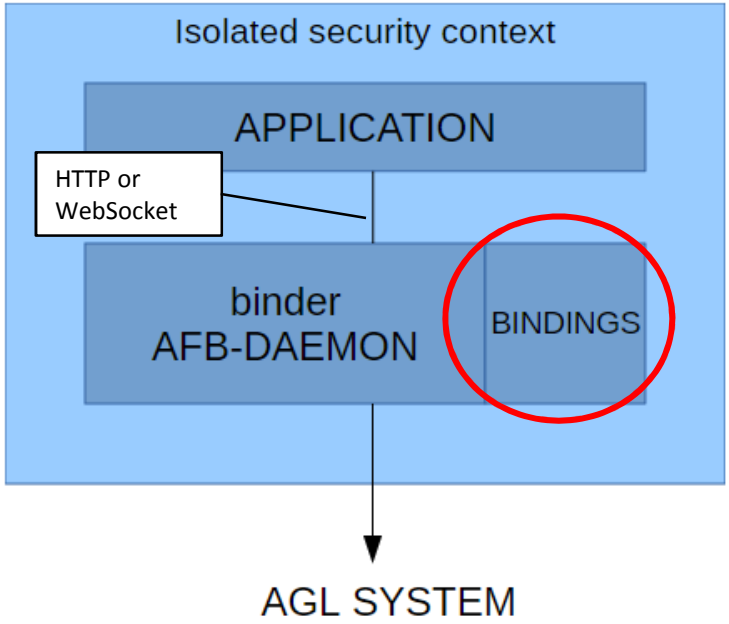
AGL AFB Binding の書き方入門

～How to write AGL Application Framework Binder Binding～

株式会社大阪エヌデーエス
エンベデッドグループ
Linuxチーム 山口洋樹

1人の満足から、社会の満足へ
株式会社大阪エヌデーエス

1. Bindingとは



図：AGL Application Framework の基本構成

引用：
http://docs.automotivelinux.org/docs/apis_services/en/dev/reference/af-binder/afb-overview.html

Application

・GUIアプリケーション
(地図、メディアプレーヤー etc..)

Binder

・アプリケーションにAPIを公開
※API：ハード・MWの機能を提供
・Webコンテナ

Binding

・Binderに追加するAPI
・実体は共有ライブラリ(.so)

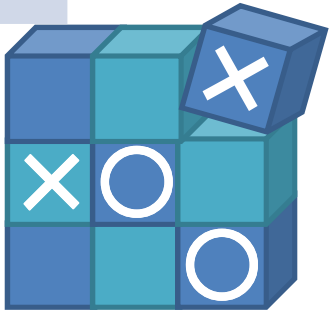
2. Bindingの書き方

・Tic-Tac-Toe (〇×ゲーム) のサンプルBindingを例に説明
・ソースコードは bindings/samples/tic-tac-toe.c

API	Verbs(Methods)	Description
tictactoe	new	新規対戦を開始する。
	play	コンピュータが〇/×をボードに書く。
	move	プレーヤーが〇/×をボードに書く。
	board	ボードの状態を確認する
	・・・	・・・

リクエスト：http://[board_ip]:[port]/api/**API/Verb**

例) http://[board_ip]:[port]/api/tictactoe/board



2.1 必要なヘッダファイル

以下のヘッダファイルは最低限必要。

bindings/samples/tic-tac-toe.c:18

```
#define _GNU_SOURCE
#include <stdio.h>
#include <string.h>
#include <json-c/json.h>

#include <afb/afb-binding.h>
```

json-c/json.h
　jsonオブジェクトを扱うための機能を含む。

afb/afb-binding.h
　Bindingが必要とする基本機能を含む。

2.2 Bindingの登録

Binder起動時にBindingを登録する。

bindings/samples/tic-tac-toe.c:600

```
/*
 * activation function for registering the binding called by afb-daemon
 */
const struct afb_binding *afbBindingV1Register(const struct afb_binding_interface *itf)
{
   afbitf = itf;         // records the interface for accessing afb-daemon /*インターフェースの保存 */
   return &binding_description;  // returns the description of the binding /* Binding情報の返却 */
}
```

・インターフェースの保存
- Binderとのインターフェースを保存する必要がある。

  ※インターフェース：
  Binderに何か依頼するときに使用する。(例：ログ出力)
- 保持するための変数ポインタは以下のように定義する。

bindings/samples/tic-tac-toe.c:28

```
const struct afb_binding_interface *afbitf;
```

・Bindingの設定と初期化（※tictactoeでは行っていない。）
- 必要に応じてBinding自身の設定や初期化処理もここで行う。

・Binding情報の返却
- 戻り値でBinding自身の情報をBinderに渡す。

2.2 Bindingの登録

bindings/samples/tic-tac-toe.c:586

```
/*
 * description of the binding for afb-daemon
 */
static const struct afb_binding binding_description =
{
   /* description conforms to VERSION 1 */
   .type= AFB_BINDING_VERSION_1,  …①
   .v1= {                                  /* fills the v1 field of the union when AFB_BINDING_VERSION_1 */
      .prefix= "tictactoe",                /* the API name (or binding name or prefix) */   …②
      .info= "Sample tac-tac-toe game",    /* short description of of the binding */        …③
      .verbs = binding_verbs               /* the array describing the verbs of the API */  …④
   }
}
```

・Binding情報の定義
① バージョン（現在はVersion1.0）
② API名（このBindingのAPI名は“tictactoe”）
③ Bindingの概要説明
④ Verbの一覧

2.2 Bindingの登録

bindings/samples/tic-tac-toe.c:570

```
/*
 * array of the verbs exported to afb-daemon
 */
static const struct afb_verb_desc_v1 binding_verbs[] = {
   /* VERB'S NAME     SESSION MANAGEMENT          FUNCTION TO CALL  SHORT DESCRIPTION */
   { ①.name= "new",   ②.session= AFB_SESSION_NONE,③.callback= new,  ④.info= "Starts a new game" },
   { .name= "play",   .session= AFB_SESSION_NONE, .callback= play,   .info= "Asks the server to play" },
   { .name= "move",   .session= AFB_SESSION_NONE, .callback= move,   .info= "Tells the client move" },
   { .name= "board",  .session= AFB_SESSION_NONE, .callback= board,  .info= "Get the current board" },
   { .name= "level",  .session= AFB_SESSION_NONE, .callback= level,  .info= "Set the server level" },
   { .name= "join",   .session= AFB_SESSION_CHECK,.callback= join,   .info= "Join a board" },
   { .name= "undo",   .session= AFB_SESSION_NONE, .callback= undo,   .info= "Undo the last move" },
   { .name= "wait",   .session= AFB_SESSION_NONE, .callback= wait,   .info= "Wait for a change" },
   { .name= NULL } /* marker for end of the array */
};
```

・Verb一覧は、各Method定義の配列
・Method定義
① Verb名：Binderから見たときのMethodの呼び名)
② セッション管理フラグ：
(設定すると、Methodリクエスト時に認証が必要となる。
認証なしの場合は AFB_SESSION_NONE を設定する。)
③ Callback関数：Methodがリクエストされたときに呼び出される関数ポインタ
④ 概要：Methodの概要説明

2.3 Methodの実装

board()関数の実装例

・概要：
ボードの情報(〇×ゲームの進行状況)を取得する。

bindings/samples/tic-tac-toe.c:338

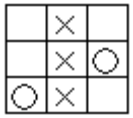
```
/*
 * get the board
 */
static void board(struct afb_req req)
{
        struct board *board;
        struct json_object *description;

        /* retrieves the context for the session */
        board = board_of_req(req);                                  # コンテキストの生成と取得
        INFO(afbitf, "method 'board' called for boardid %d", board->id);   # ログの出力

        /* describe the board */
        description = describe(board);                              # リプライの生成

        /* send the board's description */
        afb_req_success(req, description, NULL);                    # リプライの送信
}
```

2.3 Methodの実装

tictactoe/board 呼び出しと応答の例

```
$ curl http://192.168.3.61:12347/api/tictactoe/board?uuid=47a53ca3-30d8-4d38-b493-be292ca80dc3 | jq .
{
  "response": {
    "boardid": 15,
    "level": 1,
    "board": [
      " ",
      "X",
      " ",
      " ",
      "X",
      "O",
      "O",
      "X",
      " "
    ],
    "history": [
      4,
      5,
      1,
      6,
      7
    ],
    "winner": "X"
  },
  "jtype": "afb-reply",
  "request": {
    "status": "success"
  }
}
```

このボードのID



2.3.1 コンテキストの生成と取得

```
/*
 * get the board
 */
static void board(struct afb_req req)
{
        struct board *board;
        struct json_object *description;

        /* retrieves the context for the session */
        board = board_of_req(req);                                  # コンテキストの生成と取得
        INFO(afbitf, "method 'board' called for boardid %d", board->id);   # ログの出力

        /* describe the board */
        description = describe(board);                              # リプライの生成

        /* send the board's description */
        afb_req_success(req, description, NULL);                    # リプライの送信
}
};
```

2.3.1 コンテキストの生成と取得

bindings/samples/tic-tac-toe.c:307

```
/*
 * retrieves the board of the request
 */          ①                                    ②
static inline struct board *board_of_req(struct afb_req req)
{
        return afb_req_context(req, (void*)get_new_board, (void*)release_board);
}
```

・コンテキスト
  - クライアント毎(セッション毎)に関連付けられたデータ
  - tictactoe binding のコンテキストは“ボード情報”(①struct board)。

・リクエスト構造体 (②afb_req)
  - Binderから引数経由で受け取ることができる。
  - 以下の目的で使う
    - リクエスト引数の取得
    - リプライの送信
    - コンテキストの保存

2.3.1 コンテキストの生成と取得

bindings/samples/tic-tac-toe.c:307

```
/*
 * retrieves the board of the request
 */
static inline struct board *board_of_req(struct afb_req req)
{
        return afb_req_context(req, (void*)get_new_board, (void*)release_board);
}
```

①

②

- afb_req_context()の概要
  - コンテキストがない場合は２つ目の引数に指定した関数(①)が呼ばれ、その関数内でコンテキストを生成し、Binderがコンテキストを保存する。
    - get_new_board()ではボード情報のコンテキストを作成。
  - コンテキスト生成済みの場合、保存していたコンテキストを取得する。
  - ３つ目に指定した関数(②)は、セッションクローズ時に呼ばれ、コンテキストの削除(開放)を行う。

2.3.1 コンテキストの生成と取得

コンテキスト生成の例：ボード情報の生成

bindings/samples/tic-tac-toe.c:74

```
/*
 * Creates a new board and returns it.
 */
static struct board *get_new_board()
{
        /* allocation */
        struct board *board = calloc(1, sizeof *board);

        /* initialisation */
        memset(board->board, ' ', sizeof board->board);
        board->use_count = 1;
        board->level = 1;
        board->moves = 0;
        do {
                board->id = (rand() >> 2) % 1000;
        } while(board->id == 0 || search_board(board->id) != NULL);

        /* link */
        board->next = all_boards;
        all_boards = board;
        return board;
}
```

2.3.2 ログの出力

```
/*
 * get the board
 */
static void board(struct afb_req req)
{
        struct board *board;
        struct json_object *description;

        /* retrieves the context for the session */
        board = board_of_req(req);                                      # コンテキストの生成と取得
        INFO(afbitf, "method 'board' called for boardid %d", board->id);   # ログの出力

        /* describe the board */
        description = describe(board);                                  # リプライの生成

        /* send the board's description */
        afb_req_success(req, description, NULL);                        # リプライの送信
}
};
```

2.3.2 ログの出力

・ログ出力処理はマクロで定義されている
・Binderの起動オプションで出力するログのレベルを設定できる

Verbs for logging messages

The 5 logging methods are:

Macro	Verbosity	Meaning	syslog level
ERROR	0	Error conditions	3
WARNING	1	Warning conditions	4
NOTICE	1	Normal but significant condition	5
INFO	2	Informational	6
DEBUG	3	Debug-level messages	7

引用:
http://docs.automotivelinux.org/docs/apis_services/en/dev/reference/af-binder/afb-bindings-writing.html#sending-messages-to-the-log-system-1

（例）tictactoe/board リクエスト時のログ

```
INFO: method 'board' called for boardid 15 {binding tictactoe}
```

2.3.3 リプライの生成

```
/*
 * get the board
 */
static void board(struct afb_req req)
{
        struct board *board;
        struct json_object *description;

        /* retrieves the context for the session */
        board = board_of_req(req);                                      # コンテキストの生成と取得
        INFO(afbitf, "method 'board' called for boardid %d", board->id);  # ログの出力

        /* describe the board */
        description = describe(board);                                  # リプライの生成

        /* send the board's description */
        afb_req_success(req, description, NULL);                        # リプライの送信
}
};
```

2.3.3 リプライの生成

bindings/samples/tic-tac-toe.c:250

```
/*
 * get the board description
 */                                                    ①
static struct json_object *describe(struct board *board)
{
        int i;
        char w;
        struct json_object *resu, *arr;

        resu = json_object_new_object();

        json_object_object_add(resu, "boardid", json_object_new_int(board->id));
        json_object_object_add(resu, "level", json_object_new_int(board->level));

        ・・・

        return resu;
}
```

・jsonオブジェクトの作成
  - Binder間で、クエスト・リプライデータはjsonオブジェクトでやり取りする
  - tictactoe ではボード情報(①)からリプライデータを作成

2.3.4 リプライの送信

```
/*
 * get the board
 */
static void board(struct afb_req req)
{
        struct board *board;
        struct json_object *description;

        /* retrieves the context for the session */
        board = board_of_req(req);                                      # コンテキストの生成と取得
        INFO(afbitf, "method 'board' called for boardid %d", board->id);  # ログの出力

        /* describe the board */
        description = describe(board);                                  # リプライの生成

        /* send the board's description */
        afb_req_success(req, description, NULL);                        # リプライの送信
}
};
```

2.3.4 リプライの送信

include/afb/afb-req-itf.h:155

```
/*
 * Sends a reply of kind success to the request 'req'.
 * The status of the reply is automatically set to "success".
 * Its send the object 'obj' (can be NULL) with an
 * informationnal comment 'info (can also be NULL).
 *
 * For convenience, the function calls 'json_object_put' for 'obj'.
 * Thus, in the case where 'obj' should remain available after
 * the function returns, the function 'json_object_get' shall be used.
 */
static inline void afb_req_success(struct afb_req req, struct json_object *obj, const char *info)
```

・リプライの送信(「成功」の場合)
  - 作成したjsonオブジェクトをBinderに渡す。
  - info にはコメント等を設定する
    - afb_req_success_f()を使用するとinfoの出力フォーマット指定ができる
  - アプリへのリプライ送信はBinderが行う。

「失敗」を返す場合、 **afb_req_fail** や **afb_req_fail_f** を使用する。

2.3.5 リクエスト引数の取得

move()関数の実装例

・概要：
ボード上の指定したマスに○/×を書く。

bindings/samples/tic-tac-toe.c:357

```
/*
 * move a piece
 */
static void move(struct afb_req req)
{
        struct board *board;
        int i;
        const char *index;

        /* retrieves the context for the session */
        board = board_of_req(req);
        INFO(afbitf, "method 'move' called for boardid %d", board->id);

        /* retrieves the arguments of the move */
        index = afb_req_value(req, “index”);                    # リクエスト引数の取得
        i = index == NULL ? -1 : atoi(index);

        ...
}
```

index：○/×を書きたいボード上のマスの位置。

2.3.5 リクエスト引数の取得

include/afb/afb-req-itf.h:119

```
/*
 * Gets from the request 'req' the string value of the argument of 'name'.
 * Returns NULL if when there is no argument of 'name'.
 * Returns the value of the argument of 'name' otherwise.
 *
 * Shortcut for: afb_req_get(req, name).value
 */
static inline const char *afb_req_value(struct afb_req req, const char *name)
{
        return afb_req_get(req, name).value;
}
```

①　②

・リクエスト引数の取得
  - 引数の情報はリクエスト構造体(①)に格納されている
  - 2つ目の引数(②)で取り出す引数のキーを指定する

さいごに

・今回紹介した内容はBindingの書き方の一部です。
詳細はソースコード、ドキュメントを参考ください。

・今回紹介できなかった項目
- イベント通知
- 同期・非同期処理
- ビルド手順

etc…

参考資料

・Source Code

https://gerrit.automotivelinux.org/gerrit/src/app-framework-binder.git (branch: chinook)

※git clone コマンドで取得してください。

・Document

http://docs.automotivelinux.org/docs/apis_services/en/dev/reference/af-binder/afb-bindings-writing.html