Confidential B

MEDIATEK

Pluto FW 09B Enhancement Introduction

Training Course

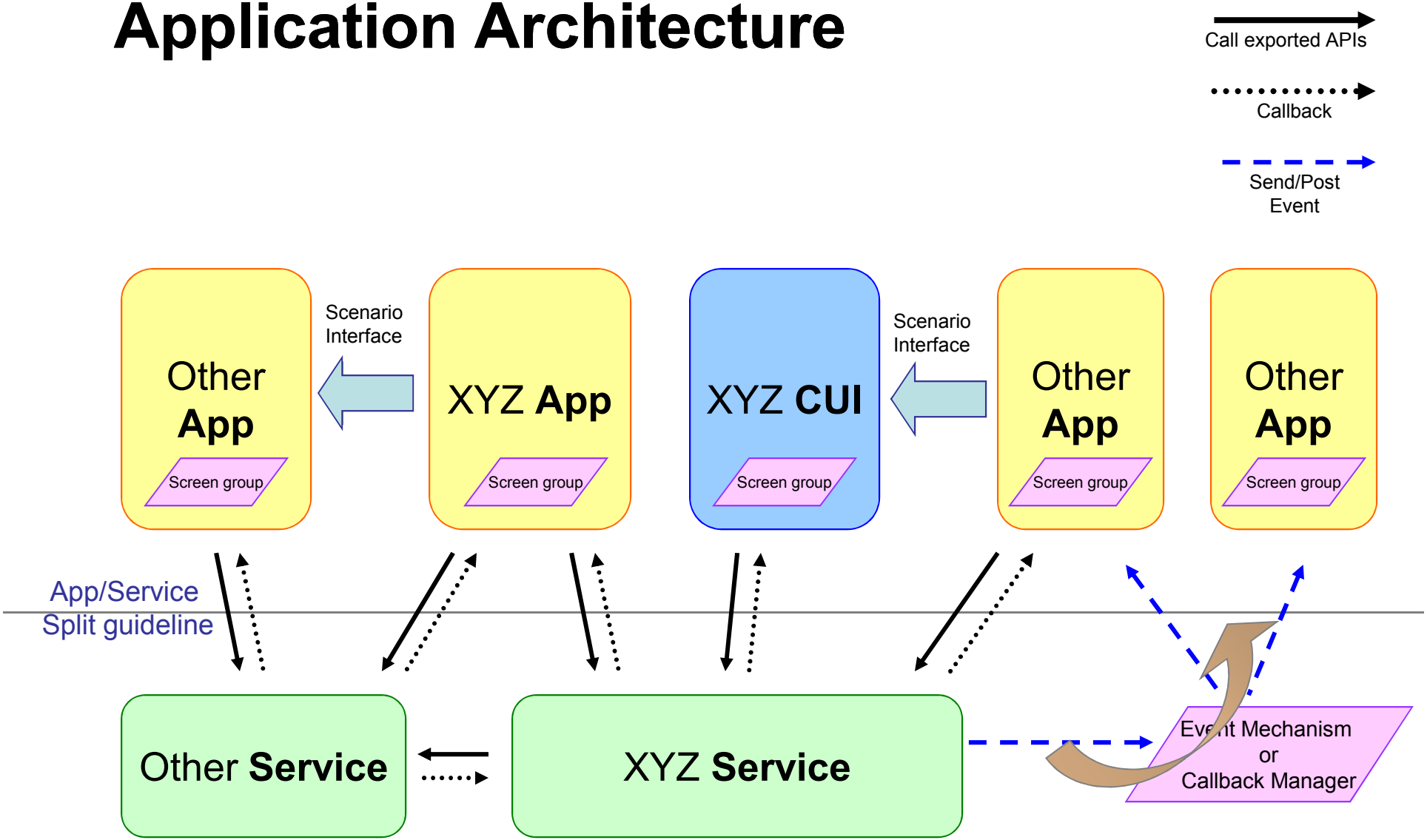
Outline

- Introduction
- Event Mechanism
- Screen Group
- Scenario Interface
- CUI
- Hierarchy ASM
- Q&A

MEDIATEK

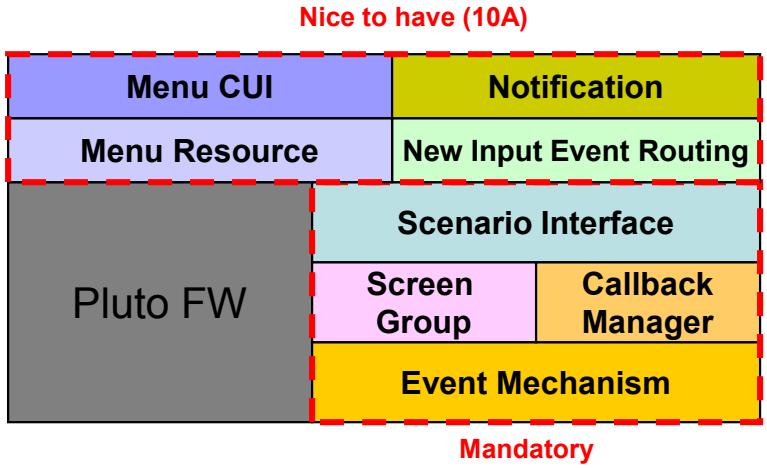
Introduction

Application Architecture



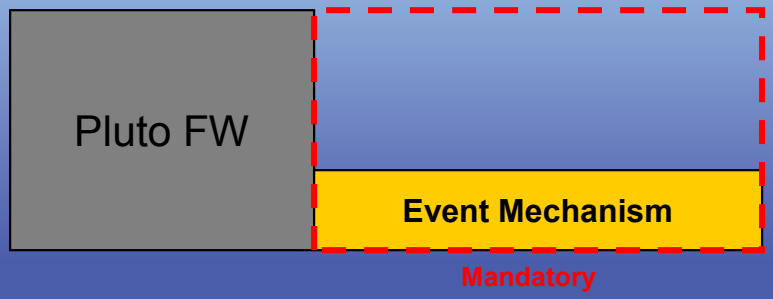
Pluto Framework Enhancement

- Mandatory (09B)
  - Screen group
  - Scenario Interface
  - Hierarchy Memory
  - 09B CUI



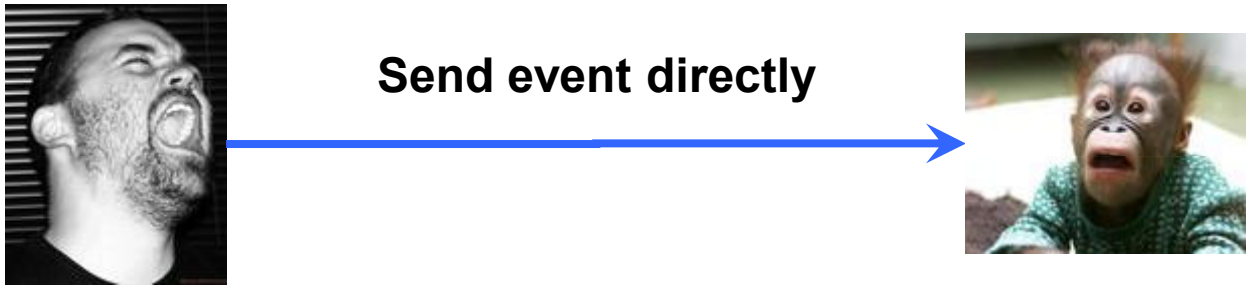
MEDIATEK

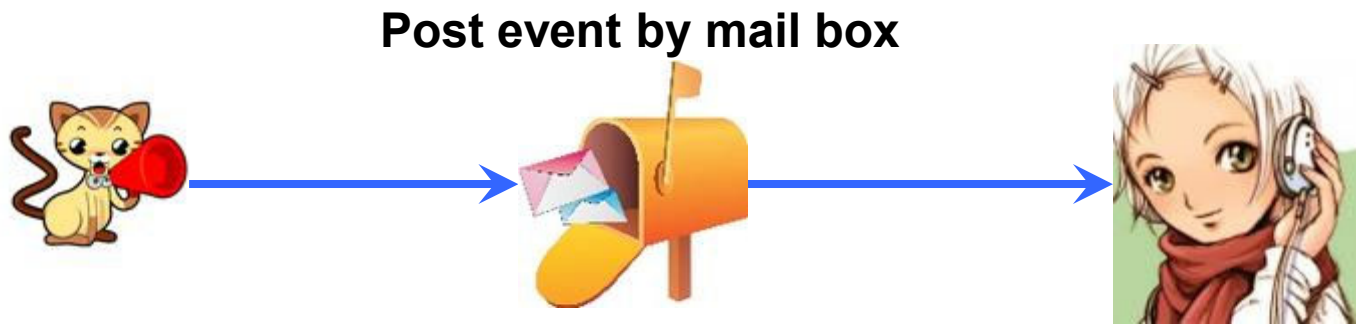
Event Mechanism



Overview

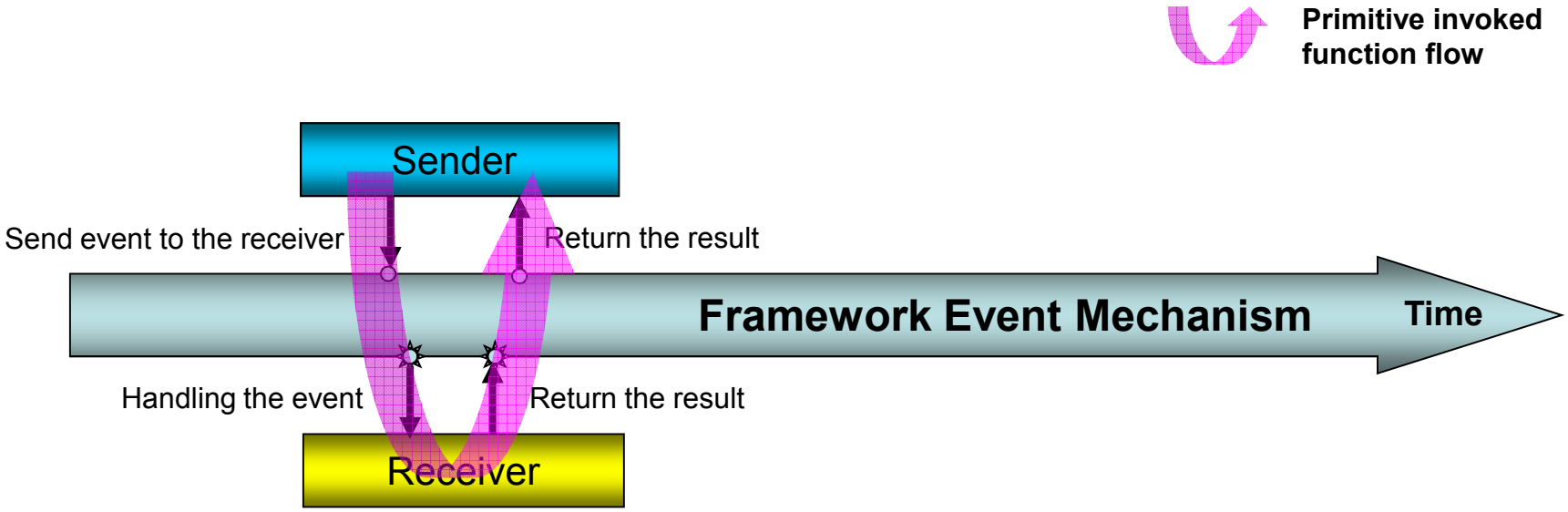
- Define event ID
- Send / Post event to the target (**proc** function)





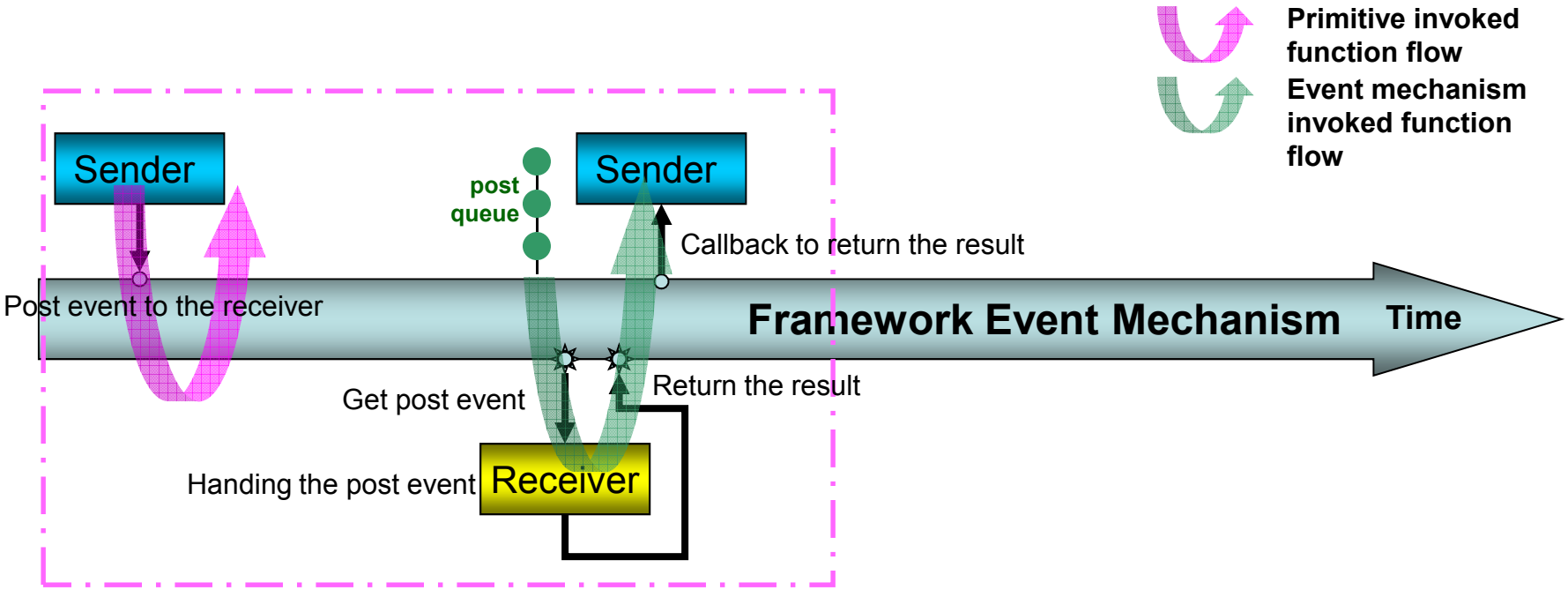
Send Event

- Caller sends event to the target (**proc** function)



Post Event

- The post event will be handled by receiver when the primitive function flow finished



Event Structure & Event ID

```
#define MMI_EVT_PARAM_HEADER \
        U16 evt_id;     \
        U16 size;        \
        void *user_data;
```

```
typedef struct
{
    MMI_EVT_PARAM_HEADER
} mmi_event_struct;
```

event's base class

```
mmi_ret (*mmi_proc_func) (mmi_event_struct *param);
```

EX: Define event ID

```
typedef enum _ucm_event_enum
{
  EVT_ID_UCM_CALL_CHANGE = (UCM_BASE+1),
  EVT_ID_UCM_MAKE_CALL_RESULT,
  ...
}ucm_evrnt_enum;
```

EX: Define event struct

```
typedef struct {
  MMI_EVT_PARAM_HEADER
  mmi_ucm_result_enum result;
} mmi_event_ucm_make_call_result_struct;
```

event's base class

EX:

```
mmi_ret receiver_proc (mmi_event_struct *evt)
{
  switch(evt->evt_id)
  {
    case EVT_ID_UCM_MAKE_CALL_RESULT:
      mmi_event_ucm_make_call_result_struct *in =
        (mmi_event_ucm_make_call_result_struct*)evt;
      ...
      break;
  }
}
```

If need detail information

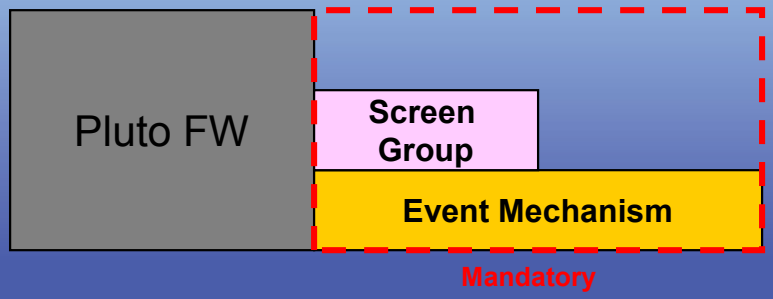
Example

```
static void mmi_ucm_make_call_result(mmi_ucm_result_enum result)
{
    mmi_ucm_result_type evt;

    /* handle cancel operation */

    MMI_EVT_STRUCT_INIT(&evt, EVT_ID_UCM_MAKE_CALL_RESULT);

    evt.result = result; /* e.g. MMI_UCM_RESULT_CALLED_PARTY_BUSY */

    /* post back result */
    mmi_frm_post_event((mmi_event_struct *)&evt, brw_proc, brw_user_data);

    /* send result directly */
    mmi_frm_send_event((mmi_event_struct *)&evt, brw_proc, brw_user_data);
}
```

Or

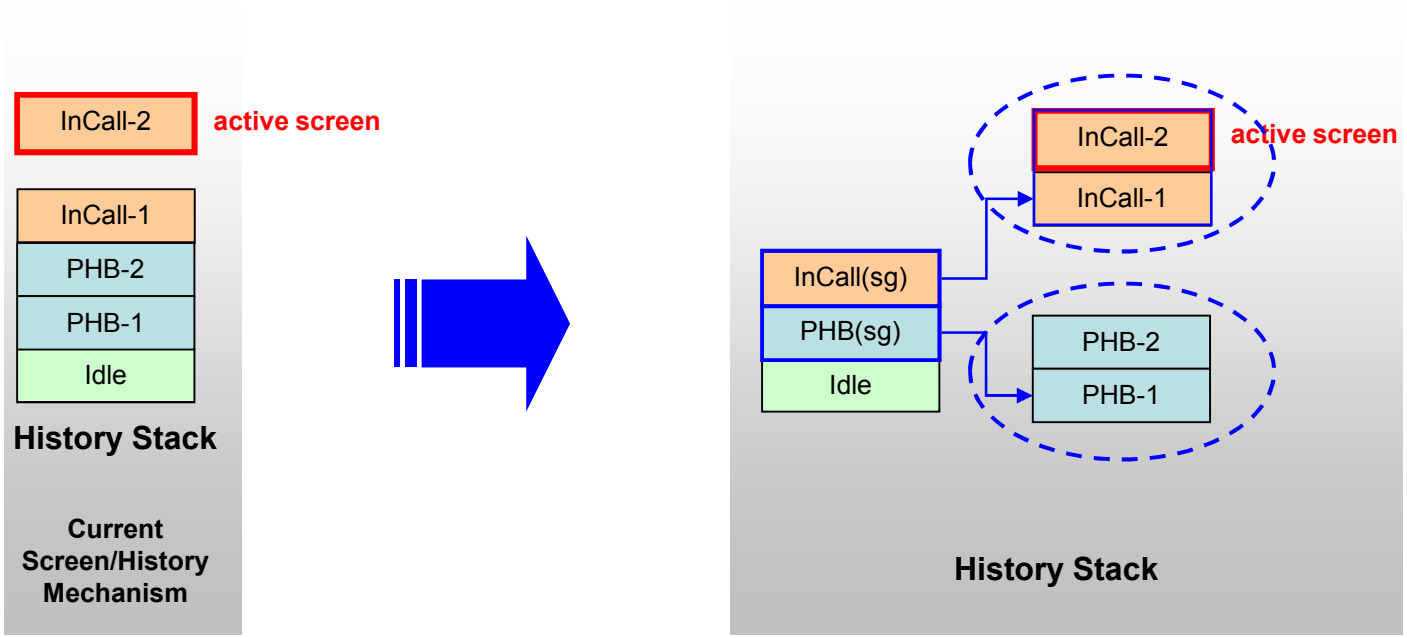
MEDIATEK

Screen Group



Concept

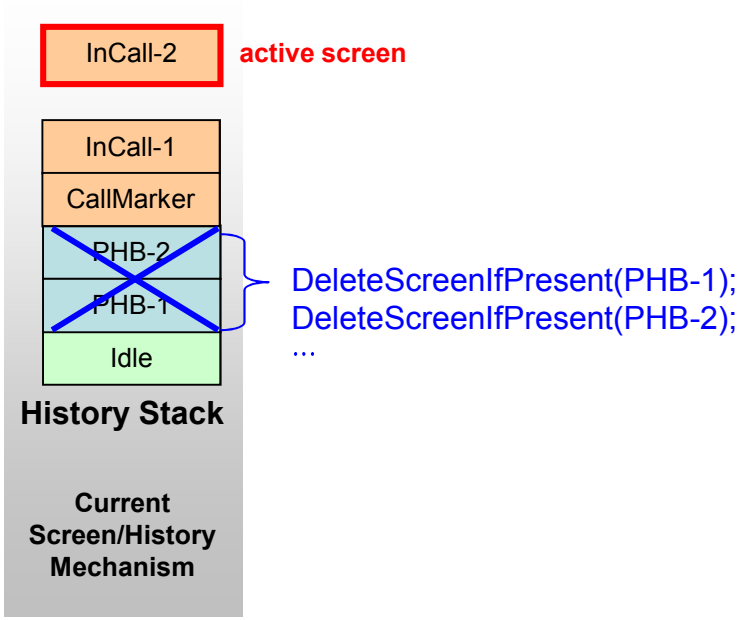
- Screen group just likes the application concept, it groups related screens together.
- Screen group ID should be unique and define from resource base



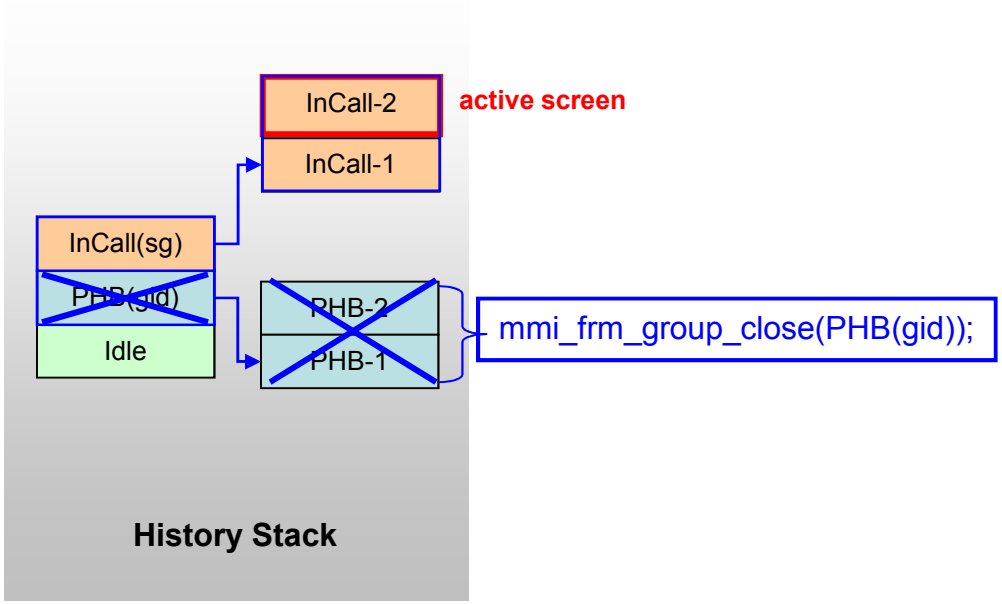
Handle the Scenario Easily – Example 1

- The users want to entry PHB again. PHB need to delete their screens

Current programming



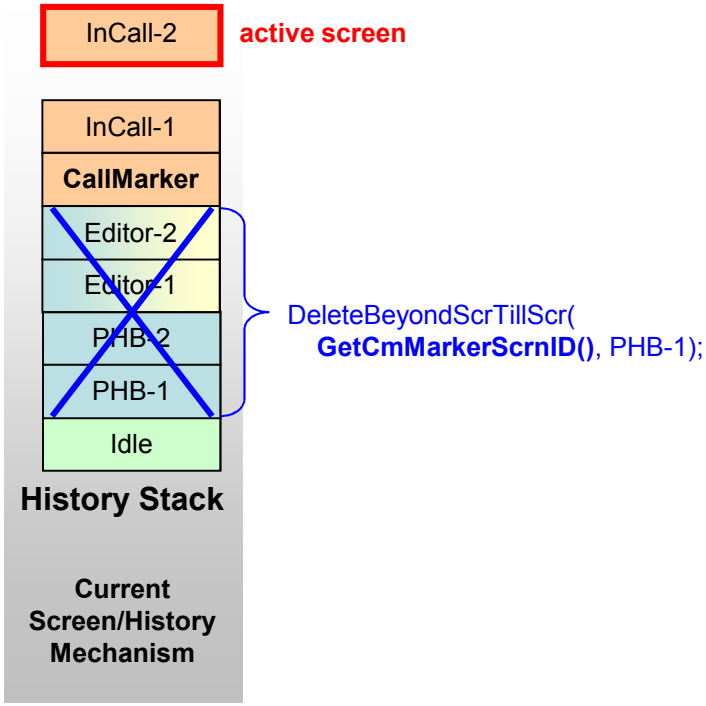
Programming with screen group



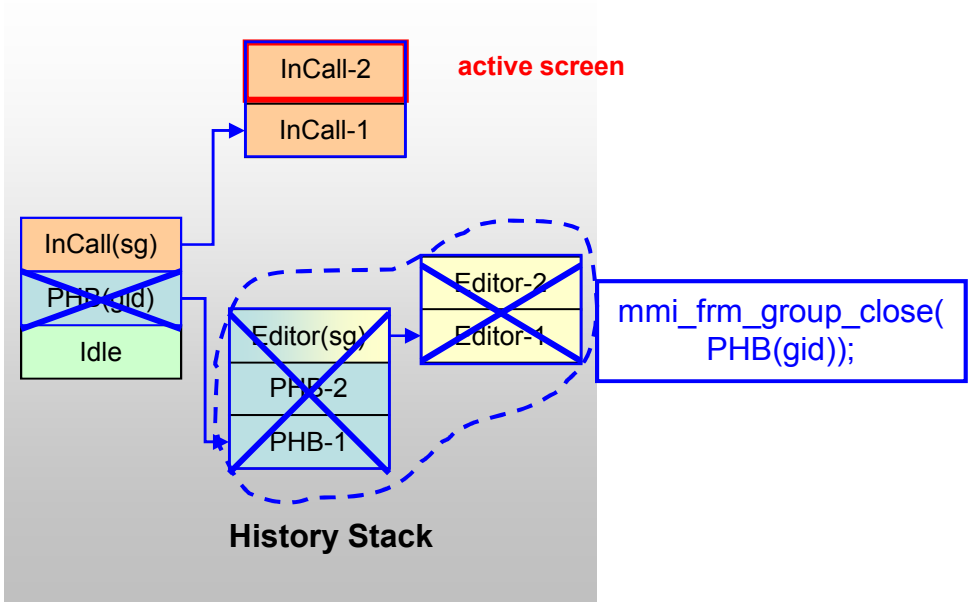
Handle the Scenario Easily – Example 2

- The users want to entry PHB again. PHB need to delete their screens (include Editor screens)

Current programming



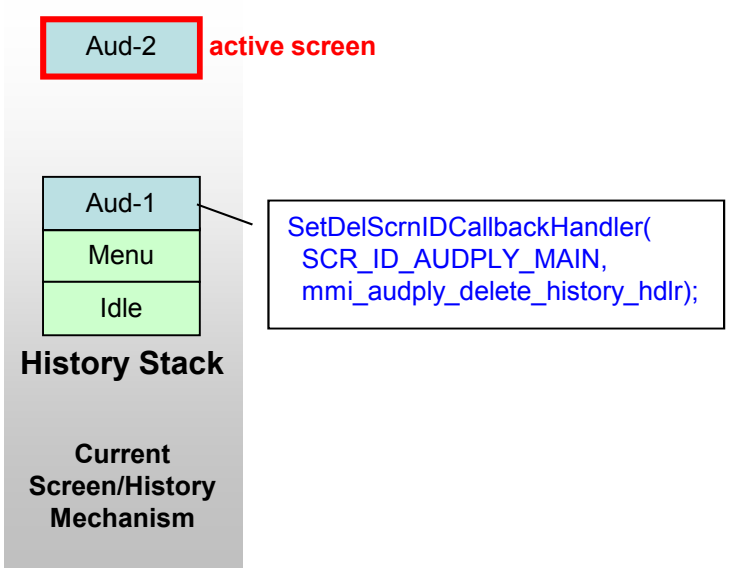
Programming with screen group



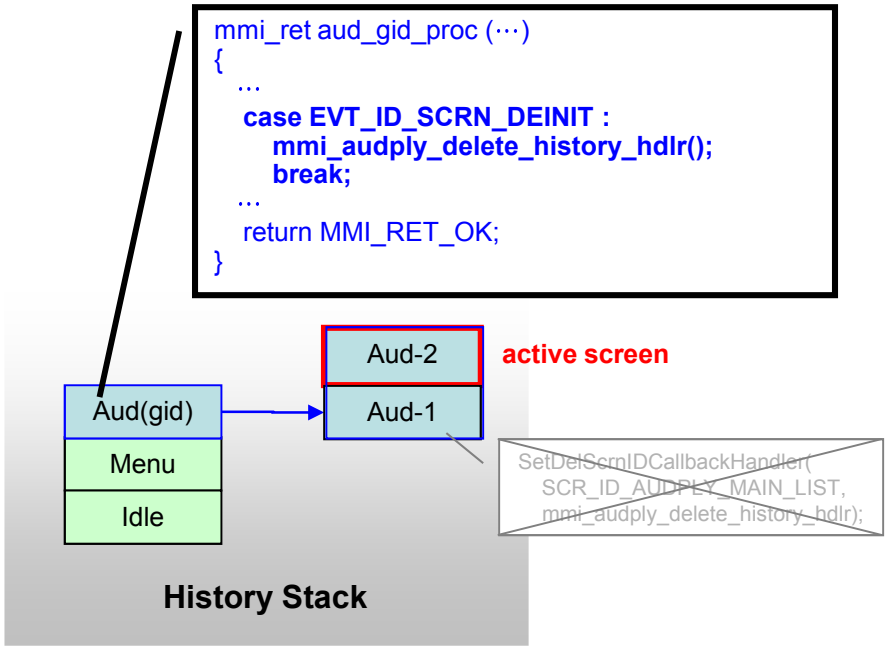
Notify App/CUI When Scenario Changes

- AudioPlayer wants to know when the users leave its application

Current programming



Programming with screen group



Features

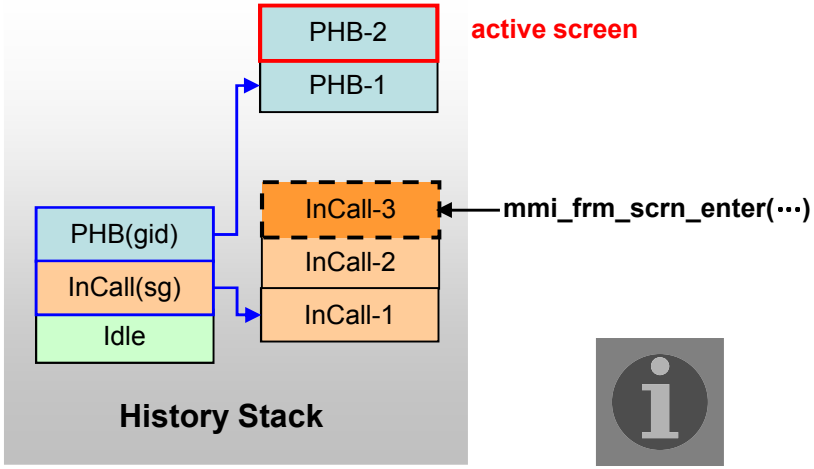
- App or CUI could handle their scenario easily
- Framework notifies App or CUI when the scenario changes
- Framework supports the notification between parent and children scenario

Example

```
typedef enum _ucm_screen_enum
{
    GRP_ID_UCM = (UCM_BASE + 1),
    GRP_ID_UCM_WITH_SOURCE,
    SCR_ID_UCM_OUTGOING,
    …
}ucm_screen_enum;

/* Phonebook uses this API to make call */
mmi_id mmi_ucm_simple_call_launch(U8 *num_uri)
{
    /* launch screen group */
    mmi_frm_group_create(GRP_ID_ROOT, GRP_ID_UCM, mmi_ucm_proc, NULL);
    mmi_frm_group_enter(GRP_ID_UCM, 0);

    mmi_ucm_call_scr_entry();     /* show screen */

    return GRP_ID_UCM;            /* return gid */
}
mmi_ret mmi_ucm_proc(mmi_event_struct *evt)
{
 switch(evt->id)
    {
    case EVT_ID_GROUP_DEINIT: // framework de-init the group
        break;
    case …
    }
    return MMI_RET_OK;
}
```

Example

```
void mmi_ucm_call_scr_entry(void)
{
    ...
    /* [entry new screen]
     * If this screen could be the active screen,
     * framework returns MMI_TRUE and executes the scenario change;
     * If this screen can't be the active screen, framework just add
     * the this screen in the scenario tree and returns MMI_FALSE.
     */
    if (mmi_frm_scrn_enter(
          GRP_ID_UCM,
          SCR_UCM_CALL_MAKE,
          exit_func,
          mmi_ucm_call_scr_entry,
          NULL))
    {
        ...    /* original code logic */
    }
}
```



Screen Group Event - Group

- EVT_ID_GROUP_ENTER
  - Enter the group node (add the group node in the scenario tree)
- EVT_ID_GROUP_ACTIVE
  - Active the group node
- EVT_ID_GROUP_INACTIVE
  - Inactive the group node
- EVT_ID_GROUP_FOCUSED
  - The group is the top active group node
- EVT_ID_GROUP_GOBACK
  - Close the active group node (go-back process)
- EVT_ID_GROUP_GOBACK_IN_END_KEY
  - Close the active group node via END key process (go-back process)
- EVT_ID_GROUP_DELETE_REQ
  - Close the inactive group node (delete process)
- EVT_ID_GROUP_DELETE_REQ_IN_END_KEY
  - Close the inactive group node via END key process (delete process)
- EVT_ID_GROUP_EXIT
  - Exit the group node (remove the group node from the scenario tree)
- EVT_ID_GROUP_DEINIT
  - Destroy the group node

Screen Group Event – Screen/Tab

- EVT_ID_SCRN_GOBACK
  - Close the active screen node (go-back process)
- EVT_ID_SCRN_GOBACK_IN_END_KEY
  - Close the active screen node via END key process (go-back process)
- EVT_ID_SCRN_DELETE_REQ
  - Close the inactive screen node (delete process)
- EVT_ID_SCRN_DELETE_REQ_IN_END_KEY
  - Close the inactive screen node via END key process (delete process)
- EVT_ID_SCRN_DEINIT
  - Destroy the screen node

- PS1:The applications could receive these events in screen leave proc
- PS2: EVT_ID_SCRN_ACTIVE & EVT_ID_SCRN_INACTIVE is reserved for future use, and take no effect on Pluto 09B.

Screen Group – Sub Items (1)

- Start group / screen / tab pages
  - MMI_ID **mmi_frm_group_create** (MMI_ID parent_id, MMI_ID group_id, mmi_proc_func proc, void *user_data);
  - MMI_ID **mmi_frm_group_enter** (MMI_ID group_id, U32 flag);
  - void **mmi_frm_scrn_first_enter** (MMI_ID parent_id, MMI_ID scrn_id, FuncPtr entry_proc, void *user_data);
  - MMI_BOOL **mmi_frm_scrn_enter** (MMI_ID parent_id, MMI_ID scrn_id, FuncPtr exit_proc, FuncPtr entry_proc, void *flag);
  - MMI_BOOL **mmi_frm_scrn_tab_enter** (MMI_ID group_id, MMI_ID scrn_id, FuncPtr exit_proc, FuncPtr entry_proc, mmi_frm_tab_struct *tab_pages_info_array, U8 tab_pages_count, U8 sel_idx);
  - MMI_BOOL **mmi_frm_scrn_tab_page_enter** (MMI_ID parent_id, MMI_ID tab_id, MMI_ID page_id, FuncPtr exit_proc, FuncPtr entry_proc, mmi_frm_scrn_type_enum scrn_type);
- Add/Insert/Replace screen
  - mmi_ret **mmi_frm_group_insert** (MMI_ID parent_id, MMI_ID base_id, mmi_frm_node_struct *new_node_info, mmi_scenario_node_flag flag);
  - mmi_ret **mmi_frm_group_replace** (MMI_ID parent_id, MMI_ID out_id, mmi_frm_node_struct *new_node_info);
  - mmi_ret **mmi_frm_scrn_insert** (MMI_ID parent_id, MMI_ID base_id, mmi_frm_node_struct *new_node_info, mmi_scenario_node_flag flag);
  - mmi_ret **mmi_frm_scrn_replace** (MMI_ID parent_id, MMI_ID out_id, mmi_frm_node_struct *new_node_info);
- mmi_frm_node_struct
  - id, entry_proc, user_data

Screen Group – Sub Items (2)

- Close group / screen [Goback/Delete screen]
  - void **mmi_frm_group_close** (MMI_ID group_id);
  - void **mmi_frm_scrn_close_active_id** (void);
    - GoBackHistory will invoke **mmi_frm_scrn_close_active_id()**
  - void **mmi_frm_scrn_close** (MMI_ID parent_id, MMI_ID scrn_id);
  - void **mmi_frm_scrn_multiple_close** (MMI_ID parent_id, MMI_ID start_scrn_id, U8 b_inc_start, U16 count, MMI_ID end_scrn_id, U8 b_inc_end);
- Set group / screen information
  - void **mmi_frm_node_info_init** (mmi_frm_node_struct *node_info);
  - mmi_ret **mmi_frm_group_set_caller** (MMI_ID group_id, MMI_ID owner_id);
  - mmi_ret **mmi_frm_group_set_proc_data** (MMI_ID group_id, mmi_proc_func proc, void *user_data);
  - mmi_ret **mmi_frm_scrn_set_leave_proc** (MMI_ID parent_id, MMI_ID scrn_id, mmi_proc_func proc);
  - mmi_ret **mmi_frm_scrn_set_user_data** (MMI_ID parent_id, MMI_ID scrn_id, void* user_data);
  - void **mmi_frm_scrn_set_active_input_buf_ptr** (U16 *input_buf_ptr);
  - mmi_ret **mmi_frm_scrn_set_attribute** (MMI_ID parent_id, MMI_ID scrn_id, mmi_scrn_attrib_enum attrib);
  - mmi_ret **mmi_frm_scrn_clear_attribute** (MMI_ID parent_id, MMI_ID scrn_id, mmi_scrn_attrib_enum attrib);
  - mmi_ret **mmi_frm_scrn_set_input_buf_mem** (MMI_ID parent_id, MMI_ID scrn_id, MemAlloc malloc_fp, MemFree mfree_fp);
  - mmi_ret **mmi_frm_scrn_tab_page_set_user_data** (MMI_ID parent_id, MMI_ID tab_id, MMI_ID page_id, void* user_data);

Screen Group – Sub Items (3)

- Query General information
  - MMI_BOOL **mmi_frm_is_in_backward_scenario** (void);

- Query group information
  - void ***mmi_frm_group_get_user_data** (MMI_ID group_id);
  - mmi_ret **mmi_frm_group_get_info** (MMI_ID group_id, mmi_frm_group_node_struct *node_info);
  - MMI_ID **mmi_frm_group_get_active_id** (void);
  - MMI_ID **mmi_frm_group_get_top_parent_group_id** (MMI_ID group_id);
  - MMI_BOOL **mmi_frm_group_is_present** (MMI_ID group_id);
  - mmi_scenario_state_enum **mmi_frm_group_get_state** (MMI_ID parent_id);
  - MMI_BOOL **mmi_frm_group_is_in_active_serial** (MMI_ID group_id);

- mmi_frm_group_node_struct
  - parent, caller, proc, priority, user_data, state

Screen Group – Sub Items (4)

- Query screen / tab-page information
  - void ***mmi_frm_scrn_get_user_data*** (MMI_ID parent_id, MMI_ID scrn_id);
  - U8* **mmi_frm_scrn_get_active_gui_buf** (void);
  - U8* **mmi_frm_scrn_get_gui_buf** (MMI_ID parent_id, MMI_ID scrn_id);
  - U16* **mmi_frm_scrn_get_active_input_buf** (void);
  - U16* **mmi_frm_scrn_get_input_buf** (MMI_ID parent_id, MMI_ID scrn_id);
  - mmi_ret **mmi_frm_scrn_get_info** (MMI_ID parent_id, MMI_ID scrn_id, mmi_frm_scrn_node_struct *node_info);
  - MMI_ID **mmi_frm_scrn_get_active_id** (void);
  - MMI_ID **mmi_frm_scrn_get_neighbor_id** (MMI_ID parent_id, MMI_ID base_id, U8 flag);
  - U32 **mmi_frm_scrn_get_count** (MMI_ID group_id);
  - MMI_ID **mmi_frm_scrn_get_top_parent_group_id** (MMI_ID parent_id, MMI_ID scrn_id);
  - MMI_BOOL **mmi_frm_scrn_is_present** (MMI_ID parent_id, MMI_ID scrn_id, mmi_scenario_node_flag flag);
  - mmi_scenario_state_enum **mmi_frm_scrn_get_state** (MMI_ID parent_id, MMI_ID scrn_id);
  - void* **mmi_frm_scrn_tab_page_get_user_data** (MMI_ID parent_id, MMI_ID tab_id, MMI_ID page_id);
  - mmi_ret **mmi_frm_scrn_tab_page_get_info** (MMI_ID parent_id, MMI_ID tab_id, MMI_ID page_id, mmi_scrn_tab_page_node_struct *node_info);
  - mmi_scenario_state_enum **mmi_frm_scrn_tab_page_get_state** (MMI_ID parent_id, MMI_ID tab_id, MMI_ID page_id);

- mmi_frm_scrn_node_struct
  - parent, entry_proc, exit_proc, leave_proc, user_data, gui_buf, gui_buf_size, input_buf, input_buf_size, state
- mmi_scrn_tab_page_node_struct
  - tab, parent, entry_proc, exit_proc, user_data, gui_buf, gui_buf_size, input_buf, input_buf_size, state

Screen Group – Sub Items (5)

- Notify to parent / caller screen group
  - mmi_ret **mmi_frm_group_send_to_caller** (MMI_ID group_id, mmi_group_event_struct *evt);
  - mmi_ret **mmi_frm_group_send_to_parent** (MMI_ID self_gid, mmi_group_event_struct *evt);
  - void **mmi_frm_group_post_to_caller** (MMI_ID self_gid, mmi_group_event_struct *evt);
  - void **mmi_frm_group_post_to_parent** (MMI_ID self_gid, mmi_group_event_struct *evt);
  - void **mmi_frm_group_post_to_caller_ex** (MMI_ID self_gid, mmi_group_event_struct *evt);
  - void **mmi_frm_group_post_to_parent_ex** (MMI_ID self_gid, mmi_group_event_struct *evt);
  - mmi_ret **mmi_frm_scrn_send_to_parent** (MMI_ID self_gid, mmi_event_struct *evt);
  - mmi_ret **mmi_frm_scrn_post_to_parent** (MMI_ID self_gid, mmi_event_struct *evt);

- mmi_group_event_struct

```
typedef struct
{
    U16 evt_id;
    U16 size;
    void* user_data;
    MMI_ID sender_id;
} mmi_group_event_struct;
```

Group Create & Enter

- Group is like application concept. We only entry the group **once**.

```
void mmi_camera_entry_screen(void)
{
    …
    EntryNewScreen(SCR_ID_CAMERA, mmi_camera_exit_screen, mmi_camera_entry_screen, …);
    guiBuffer = GetCurrGuiBuffer(SCR_ID_CAMERA);
    ……
}
```

```
void mmi_camera_entry_screen(void)
{
    …
    mmi_frm_group_create(GRP_ID_ROOT, GRP_ID_CAMERA, mmi_camera_group_proc, …);
```

```
void mmi_camera_entry_screen(void)
{
  …
  mmi_frm_group_create (GRP_ID_ROOT, GRP_ID_CAMERA, mmi_camera_group_proc, …);
  mmi_frm_group_enter (GRP_ID_CAMERA, …);
  mmi_camera_entry_screen_int();
}

void mmi_camera_entry_screen_int(void)
{
  if (mmi_frm_scrn_enter(GRP_ID_CAMERA, SCR_ID_CAMERA, …, mmi_camera_entry_screen_int, …) == MMI_FALSE)
  {
      return;
  }
  guiBuffer = mmi_frm_scrn_get_active_gui_buf();
  ……
}
```

Advance Screen Entry & Exit Function

- If you need to use user_data in screen entry before invoking mmi_frm_scrn_entry(), you need to use mmi_frm_scrn_first_enter()

```
void mmi_camera_entry_screen_int(void)
{
    //need to use screen's user_data here, but can't get in some situations.
    if (mmi_frm_scrn_enter(GRP_ID_CAMERA, SCR_ID_CAMERA, …, mmi_camera_entry_screen_int, …))
    {
        guiBuffer = mmi_frm_scrn_get_active_gui_buf();
        ……
    }
}
```

```
void mmi_camera_entry_screen(void)
{
    mmi_frm_group_create (GRP_ID_ROOT, GRP_ID_CAMERA, mmi_camera_group_proc, …);
    mmi_frm_group_enter (GRP_ID_CAMERA, …);
    mmi_frm_scrn_first_enter(GRP_ID_CAMERA, SCR_ID_CAMERA, mmi_ca
user_data);
}

void mmi_camera_entry_screen_int(mmi_scr_essential_struct *data)
{
    //could use screen's user_data here
    if (mmi_frm_scrn_enter(GRP_ID_CAMERA, SCR_ID_CAMERA, …, mmi_camera_entry_screen_int, …))
    {
        guiBuffer = mmi_frm_scrn_get_active_gui_buf();
        ……
    }
}
```

```
typedef struct {
    MMI_ID  group_id;
    MMI_ID  scrn_id;
    void    *user_data;
} mmi_scrn_essential_struct;
```

Screen Leave_proc

- The caller could receive the event in screen's leave_proc when leaving the screen
  - The function is like Del-Scrn-Callback and Destroy-Scrn-Callback
  - Received events
    - EVT_ID_SCRN_GOBACK & EVT_ID_SCRN_GOBACK_IN_END_KEY
    - EVT_ID_SCRN_DELETE_REQ & EVT_ID_SCRN_DELETE_REQ_IN_END_KEY
    - EVT_ID_SCRN_DEINIT

```
mmi_ret mmi_screen_leave_proc (mmi_event_struct *evt)
{
    switch (evt->id)
    {
    case EVT_ID_SCRN_DELETE_REQ:
    case EVT_ID_SCRN_DELETE_REQ_IN_END_KEY:
        …
        break;
    case EVT_ID_SCRN_GOBACK:
    case EVT_ID_SCRN_GOBACK_IN_END_KEY:
        …
        break;
    case EVT_ID_SCRN_DEINIT:
        …
        break;
    default:
        return MMI_RET_ERR;
    }
    return MMI_RET_ALLOW_CLOSE;
}
```

DelScrnIDCallback with **MMI_HIST_DELETE_SCREEN_TYPE**

Check the return value; Stop the delete process if return value isn't MMI_RET_ALLOW_CLOSE

DelScrnIDCallback with **MMI_HIST_EXIT_SCREEN_TYPE**

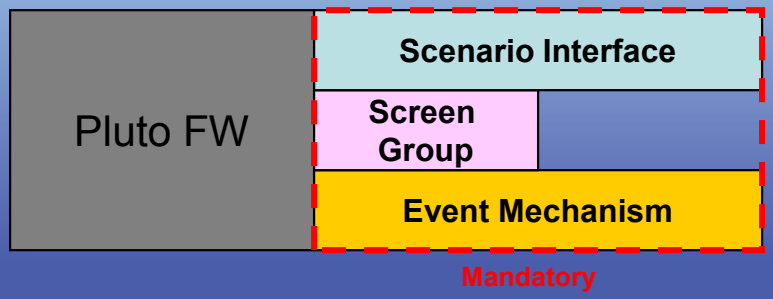
Don't check the return value

destroy_scrn_callback

Don't check the return value

Need to return value

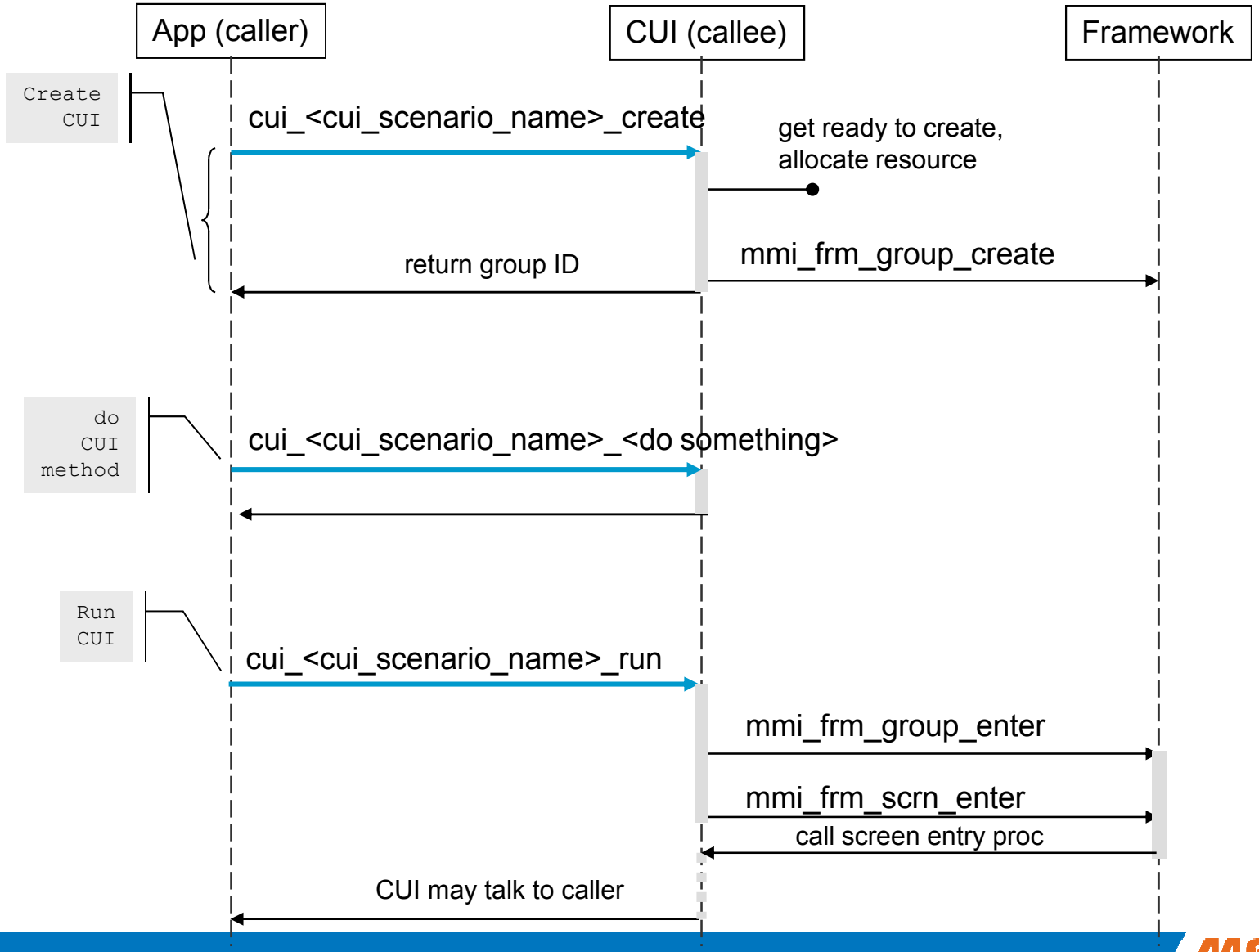
Scenario Interface



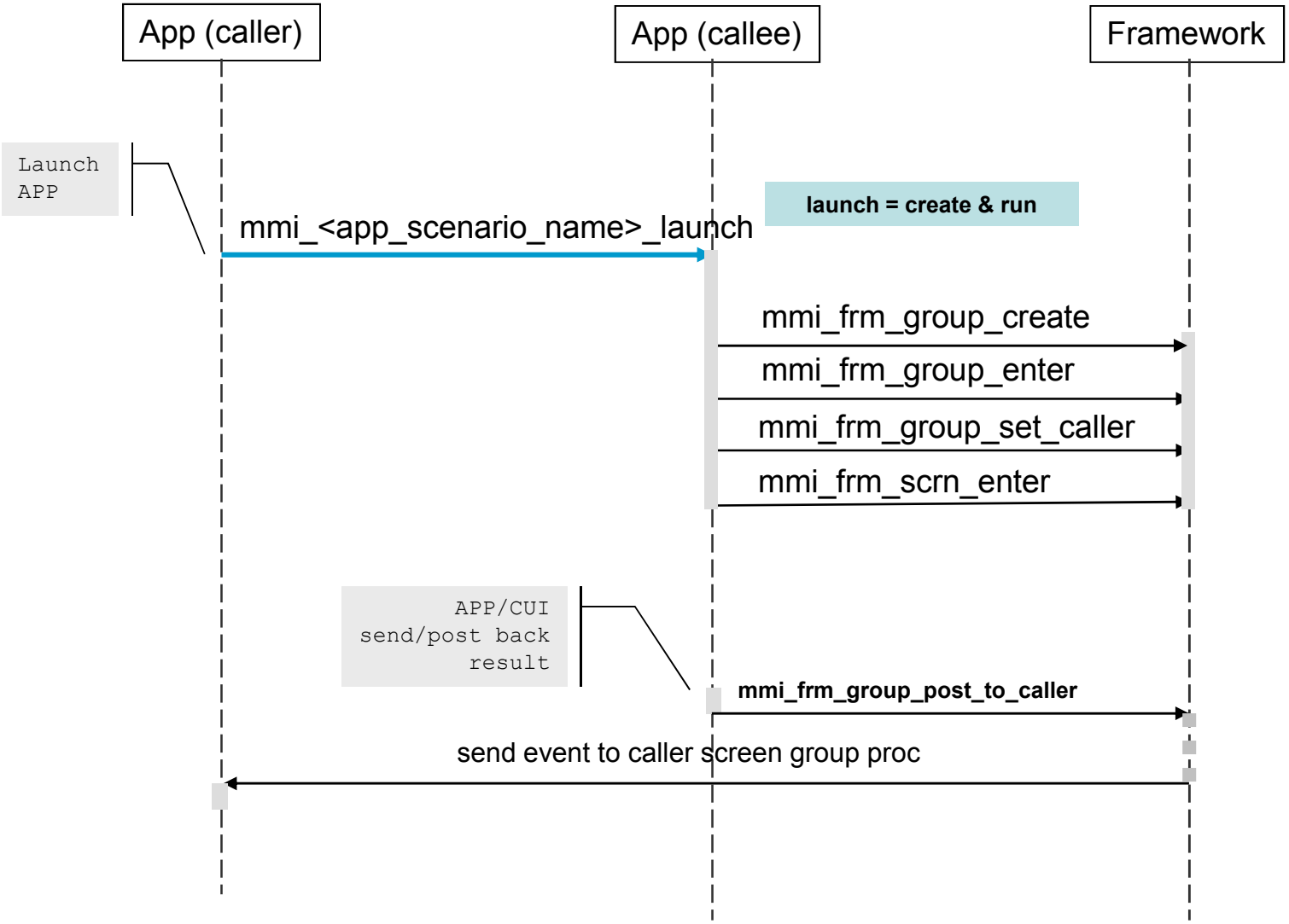
Concept

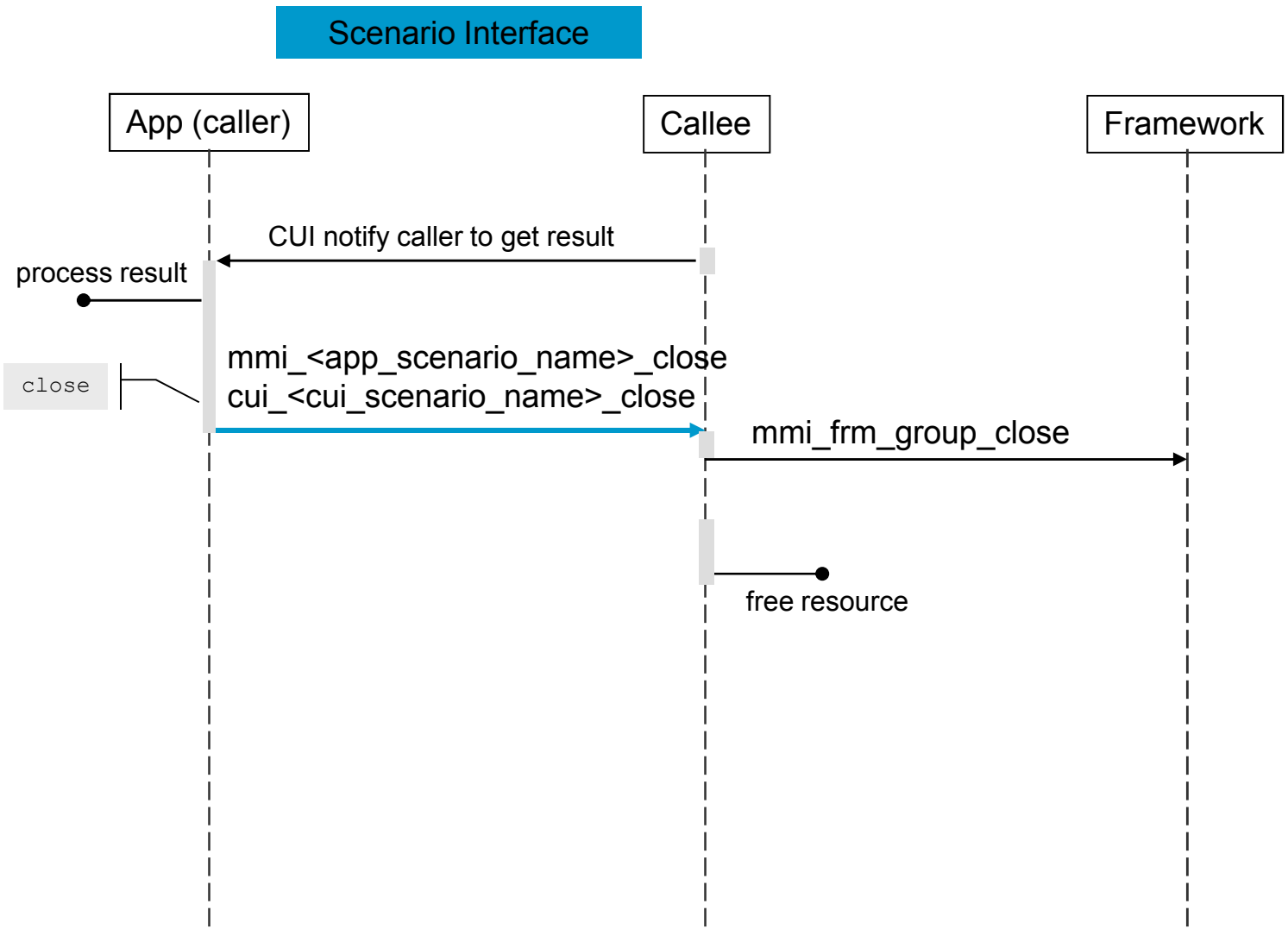
- Unify App / CUI interface
- Use group ID to communicate with App / CUI

Scenario Interface



Scenario Interface




Scenario Interface
App (caller)
Callee
Framework
CUI notify caller to get result
process result
close
mmi_<app_scenario_name>_close
cui_<cui_scenario_name>_close
mmi_frm_group_close
free resource

Scenario Interface Definition

- Simple APP launch
  - mmi_id **mmi_<app_scenario_name>_launch**(...);
- APP scenario launch w/ result back
  - mmi_id **mmi_<app_scenario_name>_launch**(mmi_id caller_gid, ...);
- APP close
  - void **mmi_<app_scenario_name>_close**(mmi_id app_gid);

- Create CUI
  - mmi_id **cui_<cui_scenario_name>_create**(mmi_id parent_gid, ...);
- Interaction w/ CUI
  - void **cui_<cui_scenario_name>_<do_something>**(mmi_id cui_gid, ...);
- Execute CUI
  - void **cui_<cui_scenario_name>_run**(mmi_id cui_gid, ...);
- Close CUI
  - void **cui_<cui_scenario_name>_close**(mmi_id cui_gid);

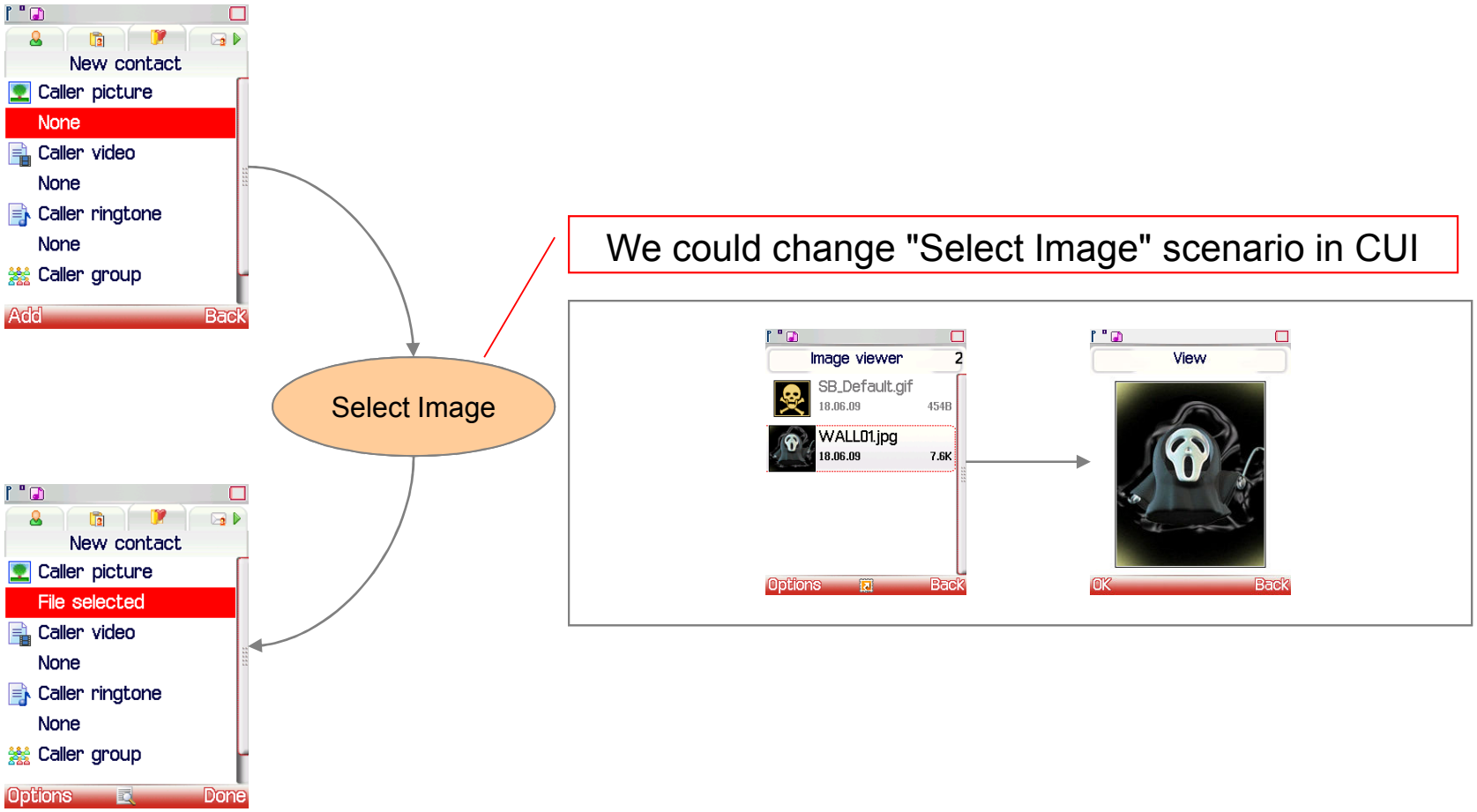
Confidential B

CUI

Introduction

- We extract common UI that different applications are interested
  - Ex. Select contact, Select tone, Select file, …
- Applications could use these CUI in the same way and we could customize CUI easily in the feature
- This change is in SW architecture, and UI and scenario should be the same with previous.

CUI Example


New contact
Caller picture
None
Caller video
None
Caller ringtone
None
Caller group
Add
Back
Select Image
New contact
Caller picture
File selected
Caller video
None
Caller ringtone
None
Caller group
Options
Done
We could change "Select Image" scenario in CUI
Image viewer
2
SB_Default.gif
18.06.09
454B
WALL01.jpg
18.06.09
7.6K
Options
Back
View
OK
Back

CUI in Pluto 09B Branch

- PHB CUI
  - Contact editor
  - One contact selector
  - Multiple contacts selector
- SMS CUI
  - SMS Sender
- FMGR CUI
  - Storage selector
  - File selector
  - Folder selector
  - Folder browser
  - File option
- BT(CM) CUI
  - Power on blue-tooth
  - Device selector
- Profile CUI
  - Tone Selector
- Camcorder (camera & video recorder) CUI
  - Camera
  - Video recorder
- Image Viewer (CUI)
- Photo Art CUI
  - Image Editor

CUI in Pluto 10A Branch

- Menu CUI
- Editor CUI
- Inline CUI
- FMGR V2.0 CUI
- Data Account CUI
- BT(BPP) CUI
- BT(BIP/OPP) CUI

Confidential B

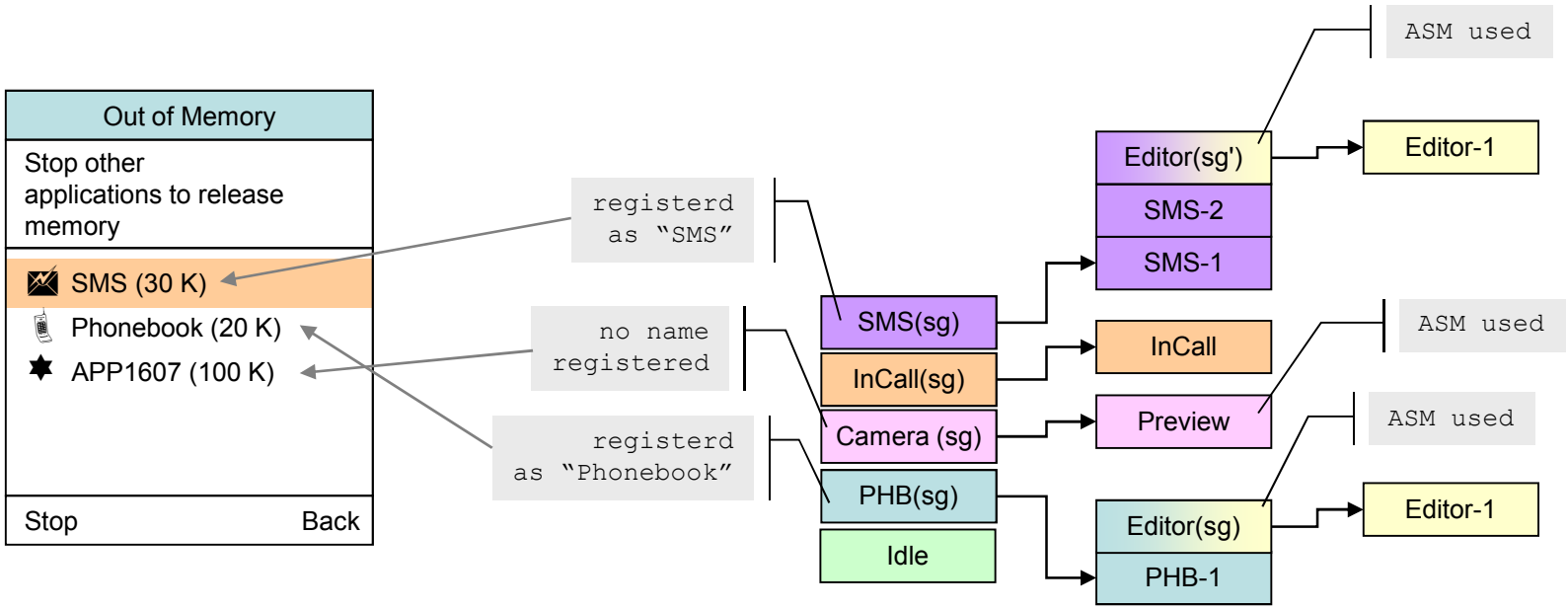
Hierarchy ASM

Background

- Screen Group and CUI are introduced to Pluto FW 09B and we have to update APP-based ASM as well to support their memory usage

Hierarchy ASM - illustration

Present root screen group registered name in Out-of-Memory screen



* If no name is registered, use default APP name.
(but this is only for error handling, it is still a bug and need to fix)

Hierarchy ASM

- APP should register its name
  - If not registered, use default name “APP x” instead, x is screen group ID
- APP will be requested to free memory if user select to free its memory in OOM (Out-of-Memory) screen
  - APP can register stop_callback as well if necessary to handle special case, e.g. have to delete screens not in screen-group
- OOM screen does not show requester itself if its child CUI requests to get more mem. and triggers OOM screen

MEDIATEK

Q&A

Q&A

- How to define group ID
  - Group ID is like screen ID, so it defines from application's resource base. CUI could use GRP_ID_GLOBAL_AUTO_GEN, and framework will generate the group ID. In all scenario tree, group ID should be unique but the same screen ID could be the different group
- How about the naming rule of App id, group id, event id?
  - Function & strucure naming
    - App - mmi_xxx
    - Service - srv_xxx
    - common UI - cui_xxx
  - App id naming
    - APP_XXX; ex. APP_PHB
    - SRV_XXX; ex. SRV_UCM
    - CUI_XXX; ex. CUI_CAMERA
  - Group id & event id naming (same with resource naming)
    - GRP_ID_XXX; ex. GRP_ID_GLOBAL_ROOT
    - EVT_ID_XXX;

Q&A

- How to revise the code of SetDelScrnIDCallbackHandler() &mmi_frm_set_destroy_scrn_callback()?
  - We provide the new API mmi_frm_scrn_set_leave_proc()
  - Framework will send the event to screen leave proc when the below situation
    - EVT_ID_SCRN_GOBACK & EVT_ID_SCRN_GOBACK_IN_END_KEY
      - When the current screen goback to previous screen
    - EVT_ID_SCRN_DELETE_REQ & EVT_ID_SCRN_DELETE_REQ_IN_END_KEY
      - When we close the screen in the history
    - EVT_ID_SCRN_DEINIT
      - When the screen will be destroyed

Application Architecture Definition

- Service
  - Data only, no UE logic and no UI
  - No customization request
  - For special purpose or reusable
  - Ex. Phonebook
    - Get the contact
- Application (App)
  - According UE implement the scenario to let the users complete the full operation
  - Ex. Phonebook, File manager, Camera, Media player
- Common UI (CUI)
  - CUI is responsible for the special purpose scenario and could be used in the different applications.
  - CUI includes several screens and it controls the screen flow and behavior
  - When the user closes App, App's CUI will be closed, too
  - Ex. Select image, Menu

What is CUI

- CUI is responsible for the special purpose scenario.
  - Ex. Select file, select the contact, get the photo from the camera, …
- CUI includes several screens and it controls the screen flow and behavior
- CUI will return the result to the caller
- CUI could be used in the different applications/scenario

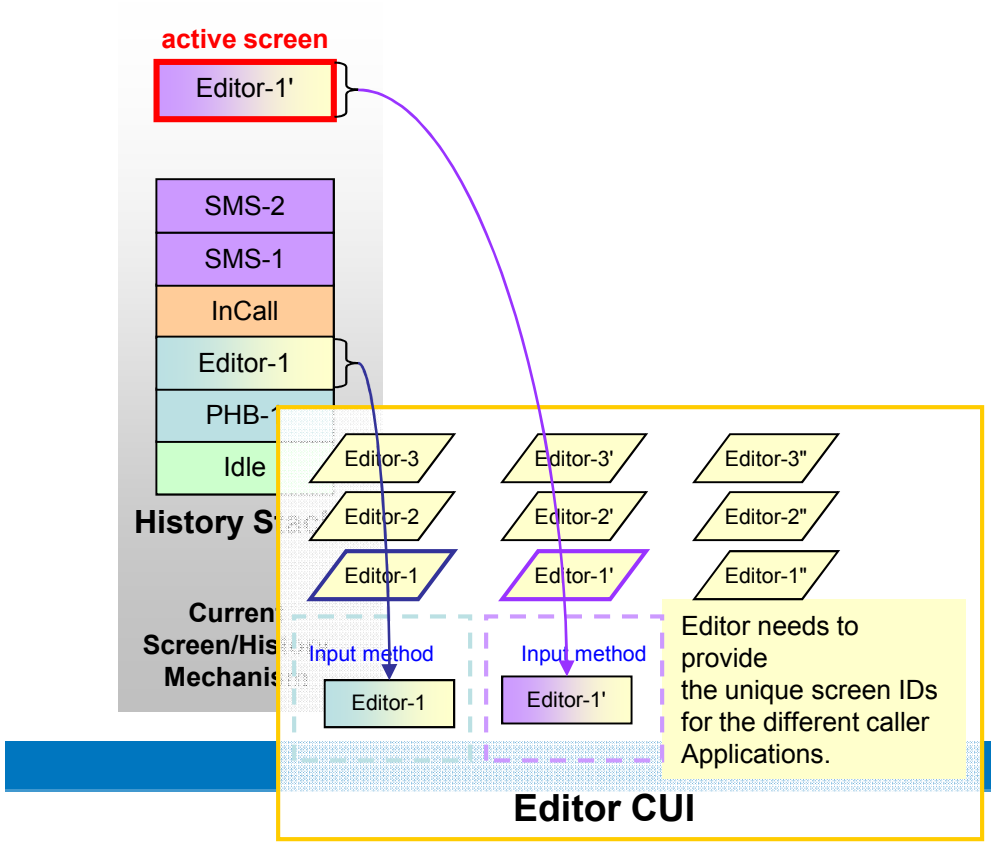
CUI Guidance

- CUI and applications should follow “scenario guidance”
  - Unify the relationship between applications and CUI
- CUI should use “screen group” to control its screens and behavior
  - CUI could support multiple instances easily
  - The caller application could easily to destroy CUI screens
- CUI should consider more than one applications may request at the same time

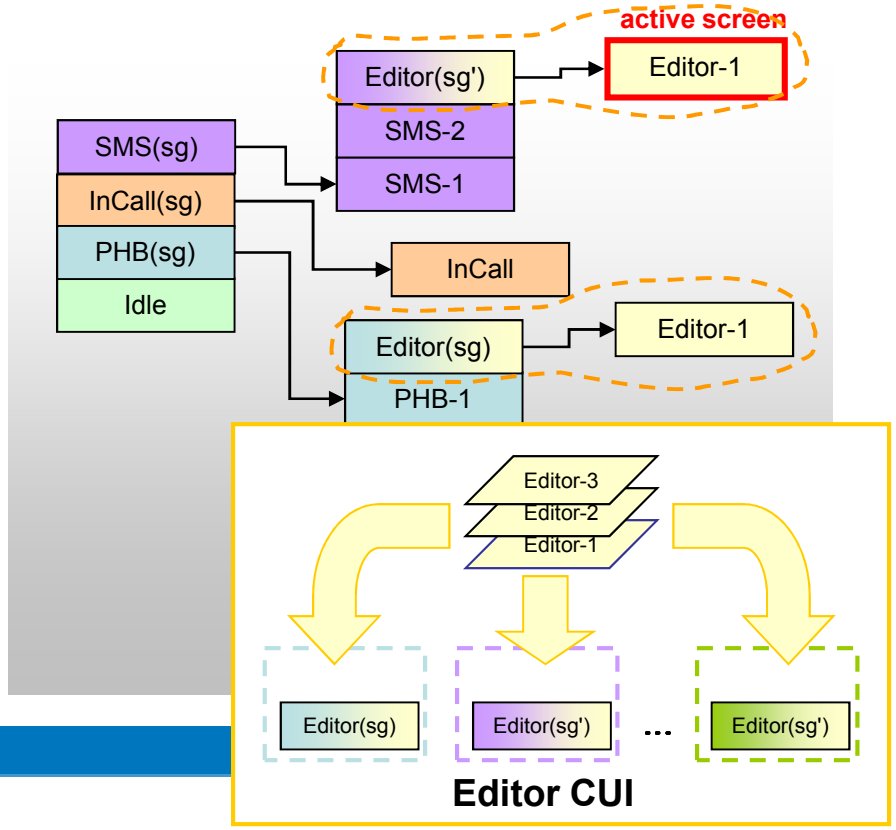
Support Same Screen ID in Different Scenario

- The applications or CUI could use the same screen IDs in the different scenario

Current programming



Programming with screen group



Example : CUI caller

- use editor in PHB contact list

```
void mmi_phb_contact_list_on_click_LSK(...)
{
    editor_gid = cui_fseditor_create(phb_list_gid);

    cui_fseditor_set_title(editor_gid, GetString(STR_SCR_PBOOK_VIEW_CAPTION),
GetImage(IMG_SCR_PBOOK_CAPTION));
    cui_fseditor_set_length(editor_gid, MMI_PHB_MAX_CONTACT_LEN);
    cui_fseditor_run(editor_gid);
}

void mmi_phb_contact_list_close(...)
{
    if (editor_gid != GRP_ID_INVALID)
    {
        cui_fseditor_close(editor_gid);
        editor_gid = GRP_ID_INVALID;
    }
}
```

Use Group ID to communicate with CUI

Example : CUI callee implementation

- CUI create, run, close

```
mmi_id cui_fseditor_create(mmi_id parent_gid)
{
 mmi_id editor_gid = GRP_ID_INVALID;

    cui_fseditor_struct *instance_data = ...; /* allocate instance memory */
    if (instance_data != NULL)
        editor_gid = mmi_frm_group_create(parent_gid, GRP_ID_AUTO_GEN,
mmi_fseditor_proc, instance_data, NULL);

    return editor_gid;
}

void cui_fseditor_run(mmi_id fseditor_gid)
{
    mmi_frm_group_enter(fseditor_gid, NULL);
    mmi_frm_scrn_enter(fseditor_gid, SCR_FSEDITOR, exit_func, entry_func, NULL);
}

void cui_fseditor_close(mmi_id fseditor_gid)
{
    /* close screen group */
    mmi_frm_group_close(fseditor_gid);
}
```

CUI's user_data

Example : CUI callee implementation

- CUI interaction

```
void cui_fseditor_set_title(mmi_id fseditor_gid, PU8 str, PU8 img)
{
    cui_fseditor_struct *editor = (cui_fseditor_struct *)
mmi_frm_group_get_user_data(fseditor_gid);

    memcpy(editor->title.str, str, mmi_ucs2strlen(str));
    editor->title.img = img;
}
```

Get CUI's user_data from group ID

Back to Q&A