WonderPro™ Fancy Device API - sample document with multiple groups using C header files

Reference Guide

80-VL700-3 C

May 17, 2011

Submit technical questions at:

https://support.cdmatech.com

Qualcomm Confidential and Proprietary

QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
U.S.A.

Contents

List of Figures

List of Tables

Revision History

Revision	Date	Description
A	Jan 2010	Initial release
B	May 2010	Updated to include advanced Doxygen markup commands to support Rev B standards
C	Sep 2010	Updated to support Rev C standards and Doxygen 1.7.0

1 Introduction

1.1 Purpose

This document describes the WonderPro™ Fancy Device (WFD) API. The WFD API provides an interface to some wonderful features.

1.2 Scope

This document is intended for software developers who will be using the WFD API.

This document provides the public interfaces necessary to use the features provided by the WFD API. A high-level overview and information on leveraging the interface functionality are also provided.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font. For example, `#include`.

Code variables appear in angle brackets. For example, `<number>`.

Commands and command variables appear in a different font. For example, **`copy a:*.* b:`**.

Test sentence 1

Test sentence 2

Test sentence 3

Test sentence 4

Test sentence 5

Test sentence 6

Test sentence 7

Test sentence 8

Test sentence 9

Test sentence 10

Parameter directions are indicated as follows:

- `[in]` indicates an input parameter.
- `[out]` indicates an output parameter.
- `[in,out]` indicates a parameter used for both input and output.

1.4 References

Reference documents, which might include Qualcomm documents and non-Qualcomm standards and resources, are listed below:

- *Application Note: Software Glossary for Customers* (CL93-V3077-1)
- *QCT Doxygen Markup Standards* (80-VP989-1)

1.5 Technical Assistance

For assistance or clarification on information in this guide, submit a case to Qualcomm CDMA Technologies at https://support.cdmatech.com.

If you do not have access to the CDMATech Support Services website, register for access or send email to support.cdmatech@qualcomm.com.

1.6 Acronyms

For definitions of terms and abbreviations, refer to the *Application Note: Software Glossary for Customers* (CL93-V3077-1). The following terms are specific to this document.

Table 1-1 Acronyms

Acronym	Definition
API	application programming interface.
AR	access rights
WFD	WonderPro™ Fancy Device

2 Functional Overview

The WFD API is a truly wonderful device.

2.1 Sample API Reference Guide

This sample API Reference Guide document was generated using the SampleHeaderFile1_multigroups_C.h, SampleHeaderFile2_multigroups_C.h, and SampleHeaderFile3_multigroups_C.h, and WFD_mainpage_multigroups_C.dox source files. This document includes example output for the marked up code items identified in the *QCT Doxygen Markup Standards* (80-VP989-1).

Figure 2-1 shows the WFD Software Arhitecture flow.

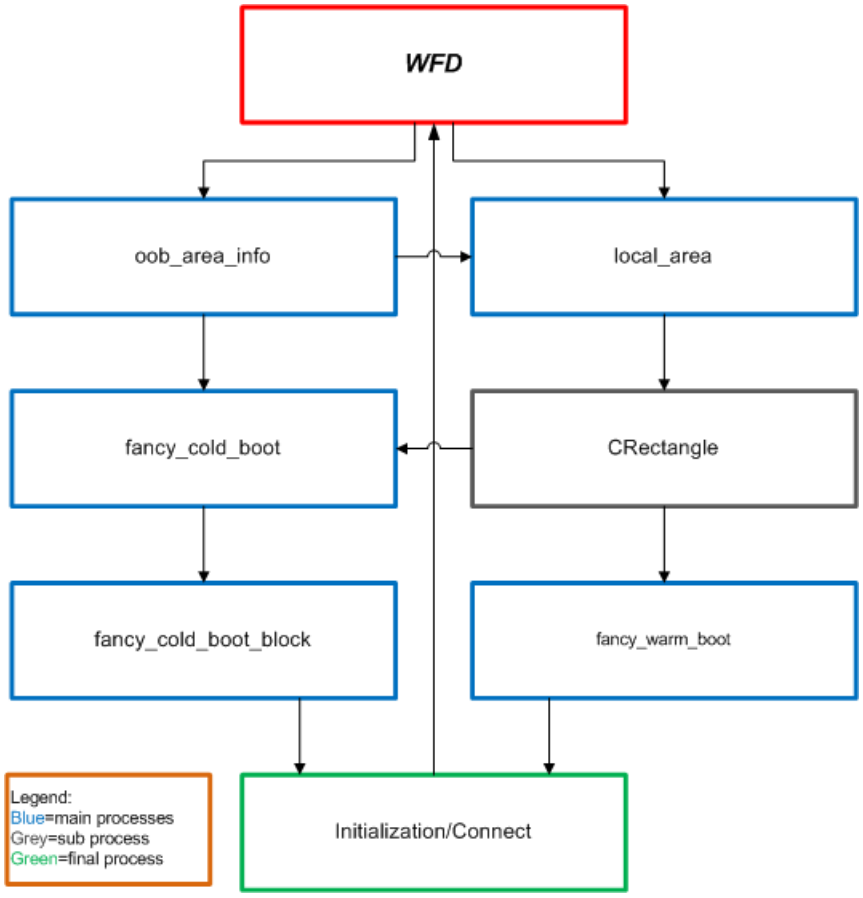


Figure 2-1 WFD Software Architecture

3 Deprecated List

Global test_process_bad_list(item_type ∗∗bad_list_msg_ptr, uint16 msg_len, bad_list_pdata_type ∗sys_bad_list, uin

Global test_process_nice_list(item_type ∗∗nice_list_msg_ptr, uint16 msg_len, nice_list_pdata_type ∗sys_nice_list, u

4 Interfaces

4.1 WFD APIs

The WFD APIs provide functionality that is truly wonderful. The WFD APIs are merely samples created to generate sample groups in the PDF output. They are not intended to be functional. These sentences all stay together in one paragraph in the PDF as part of the brief description.

Modules

- First Fancy Device API

  This is a brief description for the First Fancy Device API.
- Second Fancy Device API

  This is a brief description for the Second Fancy Device API.
- Third Fancy Device API

  This is a brief description for the third Fancy Device API.

4.2 First Fancy Device API

This is a brief description for the First Fancy Device API.

Data Structures

- struct sample1_simple_struct_s
- struct sample2_parent_struct_with_members_child_struct_as_member
- struct sample3_struct_with_two_members_and_declared_as_variable
- struct sample4_struct_with_typedef_struct_member
- struct sample5_struct_with_define_member
- struct sample6_interface_desc_t
- struct sample7_struct_with_verbatim_comment_t
- struct sample8_struct_with_union_member_t
- struct sample9_struct_sus_ad_pp_eq
- struct sample10_typedef_struct_with_member_with_bulleted_list
- struct _sample11_struct_sample_gtx_t
- struct sample12_typedefstruct

- struct sample13_typedef_packed_struct_type
- union sample14_simple_union_msg_type
- union sample15_typedef_union_u
- union sample16_union_with_struct_members_u
- struct sample17_typedef_struct_type
- struct oob_area_info
- struct local_area
- struct sample2_parent_struct_with_members_child_struct_as_member::child_struct_member

Typedefs

- typedef struct _sample11_struct_sample_gtx_t sample11_typedefstruct_sample_gtx_t

WFD Cold Boot Values

- #define FANCY_COLD_BOOT_START_BLOCK_VALUE
- #define FANCY_MAX_COLD_BOOT_END_BLOCK_VALUE
- #define FANCY_COLD_BOOT_BLOCK_VALUE_INVALID
- #define FANCY_COLD_BOOT_BLOCK_VALUE_UNASSIGNED

4.2.1 Detailed Description

This is the detailed description for the first API group.

4.2.2 Data Structure Documentation

4.2.2.1 struct sample1_simple_struct_s

Sample 1 simple struct with two members. If the brief description requires more than one sentence, it must follow the first sentence with no carriage return. If there is more information provided (more than one paragraph) for this struct, Doxygen will include with the struct's brief description, the word "more..." as a link to the additional information in the paragraph for the struct. This is an example of how to use a superscripted asterisk* in Doxygen markup.

The detailed description of the struct will be included as a lead-in paragraph to the struct's Data Fields unnumbered heading. This detailed description must be separated from the brief description for Doxygen to place this paragraph with the struct in the body of the document as shown in this example.

If a second paragraph for the detailed description is required, it must be separated from the first paragraph with a carriage return as shown here.

Data Fields

- uint8 length
- uint8 ccp [SS_MAX_LEN]

4.2.2.1.1 Field Documentation

4.2.2.1.1.1 uint8 sample1_simple_struct_s::length

Sample 1 simple struct member 1.

4.2.2.1.1.2 uint8 sample1_simple_struct_s::ccp[SS_MAX_LEN]

Sample 1 simple struct member 2.

4.2.2.2 struct sample2_parent_struct_with_members_child_struct_as_member

Sample 2 struct (parent struct) with a struct member (child struct).

The brief description (in the first paragraph) and the detailed description of the struct will be included as a lead-in paragraph to the struct's Data Fields unnumbered heading. This detailed description must be separated from the brief description for Doxygen to place this paragraph with the struct in the body of the document as shown in this example.

Note

Notice that the placement of the comment for the child struct must be inside the parent struct's closing brace to properly appear in the PDF (Need to add this change to the Rev D list). Also note that the child_struct_member info is not included in the PDF under the parent struct info, but appears later in the PDF on page 22. Dimitri is fixing this as part of SOW task 2a.

Data Fields

- int x
- int y
- struct sample2_parent_struct_with_members_child_struct_as_member::child_struct_member s

4.2.2.2.1 Field Documentation

4.2.2.2.1.1 int sample2_parent_struct_with_members_child_struct_as_member::x

Integer x in parent struct.

4.2.2.2.1.2 int sample2_parent_struct_with_members_child_struct_as_member::y

Integer y in parent struct.

4.2.2.2.1.3 struct sample2_parent_struct_with_members_child_struct_as_member::child_struct_member sample2_parent_struct_with_members_child_struct_as_member::s

Sample 2 child struct member.

4.2.2.3 struct sample3_struct_with_two_members_and_declared_as_variable

Sample 3 struct with a different construction, which includes a struct member with two members. In this example, the second usage of the struct name denotes that a variable of the same name will be created.

Data Fields

- struct StructChildMember StructChildMember
- SResult(∗ FCN_0)(uint32 leb_idx, SDeviceHandle ∗h, uint32 u1)
- SResult(∗ FCN_1)(uint32 leb_idx, SDeviceHandle ∗h, uint32 u1, uint32 u2)

4.2.2.3.1 Field Documentation

4.2.2.3.1.1 struct StructChildMember sample3_struct_with_two_members_and_declared_as_variable::StructChildMember

Sample 3 struct with two struct members and struct declared as variable.

4.2.2.3.1.2 SResult(∗ sample3_struct_with_two_members_and_declared_as_variable::FCN_0)(uint32 leb_idx, SDeviceHandle ∗h, uint32 u1)

Sample 3 struct member 1.

4.2.2.3.1.3 SResult(∗ sample3_struct_with_two_members_and_declared_as_variable::FCN_1)(uint32 leb_idx, SDeviceHandle ∗h, uint32 u1, uint32 u2)

Sample 3 struct member 2.

4.2.2.4 struct sample4_struct_with_typedef_struct_member

Sample 4 struct with a typedef struct member.

Public Types

- typedef struct sample4_typedef_member_type sample4_typedef_member

Data Fields

- int a
- int b

4.2.2.4.1 Member Typedef Documentation

4.2.2.4.1.1 typedef struct sample4_typedef_member_type sample4_struct_with_typedef_struct_member::sample4_typedef_member

Sample 4 struct member 1.

4.2.2.4.2 Field Documentation

4.2.2.4.2.1 int sample4_struct_with_typedef_struct_member::a

Sample 4 struct member 2.

4.2.2.4.2.2 int sample4_struct_with_typedef_struct_member::b

Sample 4 struct member 3.

4.2.2.5 struct sample5_struct_with_define_member

Sample 5 struct with a define member. Note that in order to include the information for the define member, one must use the as shown below. Also note that Doxygen will list the items first under separate section entitled "Friends And Related Function Documentation". A [related] tag is also placed next to its title.

Data Fields

- int a
- int b
- uint8_t number

Related Functions

(Note that these are not member functions.)

- SAMPLE_DEFINE_INSIDE_A_STRUCT

4.2.2.5.1 Friends And Related Function Documentation

4.2.2.5.1.1 SAMPLE_DEFINE_INSIDE_A_STRUCT [related]

Sample 5 struct member 3.

4.2.2.5.2 Field Documentation

4.2.2.5.2.1 int sample5_struct_with_define_member::a

Sample 5 struct member 1.

4.2.2.5.2.2 int sample5_struct_with_define_member::b

Sample 5 struct member 2.

4.2.2.5.2.3 uint8_t sample5_struct_with_define_member::number

Sample 5 struct member 4.

4.2.2.6 struct sample6_interface_desc_t

Sample 6 struct. Interface descriptor structure.

Data Fields

- if_control_msg_fn control_msg
- alt_interface_desc_t ∗ alt_ifs
- juint8_t alt_if_num
- juint8_t alt_if_curr
- juint8_t ∗ extra_descriptor
- juint32_t extra_descriptor_size
- juint8_t number
- juint8_t if_class
- juint8_t if_subclass
- juint8_t if_protocol
- juint8_t if_string

Related Functions

(Note that these are not member functions.)

- UWD_UNDEFINED_INTERFACE

4.2.2.6.1 Friends And Related Function Documentation

4.2.2.6.1.1 UWD_UNDEFINED_INTERFACE [related]

Sample 6 struct member 7 with value (0xFF).

4.2.2.6.2 Field Documentation

4.2.2.6.2.1 if_control_msg_fn sample6_interface_desc_t::control_msg

Sample 6 struct member 1.

4.2.2.6.2.2 alt_interface_desc_t∗ sample6_interface_desc_t::alt_ifs

Sample 6 struct member 2.

4.2.2.6.2.3 juint8_t sample6_interface_desc_t::alt_if_num

Sample 6 struct member 3.

4.2.2.6.2.4 juint8_t sample6_interface_desc_t::alt_if_curr

Sample 6 struct member 4.

4.2.2.6.2.5 juint8_t∗ sample6_interface_desc_t::extra_descriptor

Sample 6 struct member 5.

4.2.2.6.2.6 juint32_t sample6_interface_desc_t::extra_descriptor_size

Sample 6 struct member 6.

4.2.2.6.2.7 juint8_t sample6_interface_desc_t::number

Sample 6 struct member 8.

4.2.2.6.2.8 juint8_t sample6_interface_desc_t::if_class

Sample 6 struct member 9.

4.2.2.6.2.9 juint8_t sample6_interface_desc_t::if_subclass

Sample 6 struct member 10.

4.2.2.6.2.10 juint8_t sample6_interface_desc_t::if_protocol

Sample 6 struct member 11.

4.2.2.6.2.11 juint8_t sample6_interface_desc_t::if_string

Sample 6 struct member 12.

4.2.2.7 struct sample7_struct_with_verbatim_comment_t

Sample 7 struct with verbatim comment block.

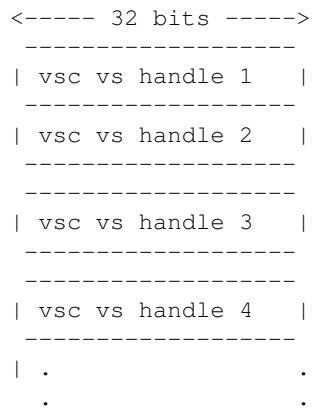
Data Fields

- uint32_t ∗ buffer
- uint32_t buffer_size
- const uint8_t ∗ pbuf
- uint32_t ∗ ret_total_size

4.2.2.7.1 Field Documentation

4.2.2.7.1.1 uint32_t∗ sample7_struct_with_verbatim_comment_t::buffer

Sample 7 struct member 1.



4.2.2.7.1.2 uint32_t sample7_struct_with_verbatim_comment_t::buffer_size

Sample 7 struct member 2.

4.2.2.7.1.3 const uint8_t* sample7_struct_with_verbatim_comment_t::pbuf

Sample 7 struct member 3.

4.2.2.7.1.4 uint32_t* sample7_struct_with_verbatim_comment_t::ret_total_size

Sample 7 struct member 4.

4.2.2.8 struct sample8_struct_with_union_member_t

Sample 8 struct with a union member that has members. Note that the comment for the union does not show up in PDF. Dimitri fixing as part of SOW task. Also note that when a data structure (struct, typedef, union, class) are used as members, do not include the brief command in their comment - need to add this change to the Rev D list.

Data Fields

- uint32_t open_id
- char * Data
- union {
  swu_vs_open_full_control_t full_control
  swu_vs_open_passive_control_t passive_control
  } u

4.2.2.8.1 Field Documentation

4.2.2.8.1.1 uint32_t sample8_struct_with_union_member_t::open_id

Sample 8 struct member 1:

- #SWU_VS_OPENID

- #SWU_VS_CLOSEID

4.2.2.8.1.2 char* sample8_struct_with_union_member_t::Data

Sample 8 struct member 2.

4.2.2.8.1.3 swu_vs_open_full_control_t sample8_struct_with_union_member_t::full_control

Sample 8 struct union member 1. This allows data to be exchanged across SWU controls.

4.2.2.8.1.4 swu_vs_open_passive_control_t sample8_struct_with_union_member_t::passive_control

Sample 8 struct union member 2. This allows data to be exchanged to manage SWU passive controls.

4.2.2.8.1.5 union { ... } sample8_struct_with_union_member_t::u

Sample8 union with members inside a struct.

4.2.2.9 struct sample9_struct_sus_ad_pp_eq

Sample 9 struct with a union member that has a struct member.

Data Fields

- uint32_t param_id
- union {
  struct sus_aud_pp_eq_enable enable
  } u

4.2.2.9.1 Field Documentation

4.2.2.9.1.1 uint32_t sample9_struct_sus_ad_pp_eq::param_id

Sample 9 struct member 1.

4.2.2.9.1.2 struct sus_aud_pp_eq_enable sample9_struct_sus_ad_pp_eq::enable

Sample 9 struct union member 1.

4.2.2.9.1.3 union { ... } sample9_struct_sus_ad_pp_eq::u

Sample 9 struct member 2.

4.2.2.10 struct sample10_typedef_struct_with_member_with_bulleted_list

Sample 10 typedef struct with member with a bulleted list in comment. In this example, the uProfiling-Level name is inside a LaTeX label command to create a hyperlink. Note that in this type of comment, a backslash n should be used prior to the bulleted list.

Data Fields

- uint32_t one_level

4.2.2.10.1 Field Documentation

4.2.2.10.1.1 uint32_t sample10_typedef_struct_with_member_with_bulleted_list::one_level

First level:

- 0 – Comment for first item in bulleted list.
- 1 – Comment for second item in bulleted list.
- 2 – Comment for third item in bulleted list.

4.2.2.11 struct _sample11_struct_sample_gtx_t

Sample 11 typedef struct with a struct member that has an enum and a union member. Note that the comment for struct member 3 (enum) and struct member 4 (union) do not show up in PDF under this typedef struct as it should. In fact, they don't show up anywhere in the PDF. Dimitri fixing as part of SOW task 2f.

Data Fields

- void * main_gtx
- gtx_appgtx_t app_gtx
- struct {
  enum { **GTX_READY**, **GTX_RESP_SENDING**, **GTX_RESP_SENT** }
  enum
  _sample11_struct_sample_gtx_t:: { ... } state
  lbool_t **not_pending**
  lbool_t **sending_zldp**
  void * **buffer**
  lbint32_t **size**
  lbint32_t **bytes_sent**
  } **sample11_typedefstruct_encap_response**

- union {
  gtx_calling_t **basic**
  led_calling_t **acm**
  leb_calling_t **ecm**
  gex_calling_t **obex**
  } sample11_union_gtx

4.2.2.11.1 Field Documentation

4.2.2.11.1.1 void* _sample11_struct_sample_gtx_t::main_gtx

Sample 11 typedef struct member 1.

4.2.2.11.1.2 gtx_appgtx_t _sample11_struct_sample_gtx_t::app_gtx

Sample 11 typedef struct member 2.

4.2.2.11.1.3 enum { ... } _sample11_struct_sample_gtx_t::state

Sample 11 typedef struct member 3 with enum.

4.2.2.11.1.4 union { ... } _sample11_struct_sample_gtx_t::sample11_union_gtx

Sample 11 typedef struct union member 4.

4.2.2.12 struct sample12_typedefstruct

Sample 12 typedef struct with a function. Note that the comment for the struct member does not show up in PDF under this typedef struct as it should. Instead, it shows up under the Variable Documentation section in the PDF on page 23. Dimitri fixing as part of SOW task 2a.

Data Fields

- sample12_GEXResult(∗ reinit)(gex_t ∗_pif, const gex_format_t ∗in_format_ptr, gex_format_t ∗out_format_ptr, gex_buf_t ∗info_ptr)

4.2.2.12.1 Field Documentation

4.2.2.12.1.1 sample12_GEXResult(∗ sample12_typedefstruct::reinit)(gex_t ∗_pif, const gex_format_t ∗in_format_ptr, gex_format_t ∗out_format_ptr, gex_buf_t ∗info_ptr)

Sample 12 function in a typedef struct.

Parameters

in	*_pif*	Pointer to the library object.
in,out	*info_ptr*	Initialization information.

Returns

Indication of success or failure.

Dependencies

The gex_getsize_f() and gex_init_f() functions must have been executed, and memory must have been allocated.

Comments

Sample comment 1.
Sample comment 2.
Sample comment 3. This sample comment is intended to be very long so that it wrap the second sentence all the way to the next line.

Errors

Sample error 1.
Sample error 2.
Sample error 3. This sample error is intended to be very long so that it wrap the second sentence all the way to the next line.

Description

Sample description that is very very long so that it will wrap onto the next line.

Important:

Sample important statement 1.

Sample important statement 2. This statement is very long so that it will wrap onto the next line.

Sample important statement 3. This statement includes a bulleted list:

- List item 1.
- List item 2.
- List item 3 which includes another list:
  1. Sublist item 1.
  2. Sublist item 2.

4.2.2.13 struct sample13_typedef_packed_struct_type

Sample 13 typedef packed struct with members and struct members. This is an example of a typedef packed struct with three members and two child struct members.

Data Fields

- boolean valid_flg
- uint32 precedence
- uint32 mean
- struct {
  lb_arcid_T arcid
  } **sample13_typedefstruct_arc_reestab_reestab_params**

- struct {
  lb_nonarc_id_T lb_nonarc_id
  } **sample13_struct_typedefstruct_nonarc_reestab_params**

4.2.2.13.1 Field Documentation

4.2.2.13.1.1 boolean sample13_typedef_packed_struct_type::valid_flg

Sample 13 typedef struct member 1.

4.2.2.13.1.2 uint32 sample13_typedef_packed_struct_type::precedence

Sample 13 typedef struct member 2.

4.2.2.13.1.3 uint32 sample13_typedef_packed_struct_type::mean

Sample 1e typedef struct member 3.

4.2.2.13.1.4 lb_arcid_T sample13_typedef_packed_struct_type::arcid

Sample 13 typedef struct member 4.

4.2.2.13.1.5 lb_nonarc_id_T sample13_typedef_packed_struct_type::lb_nonarc_id

Sample 13 typedef struct member 5.

4.2.2.14 union sample14_simple_union_msg_type

Sample 14 simple union for a C function.

Data Fields

- uint32 dummy

4.2.2.14.1 Field Documentation

4.2.2.14.1.1 uint32 sample14_simple_union_msg_type::dummy

Sample 14 union member.

4.2.2.15 union sample15_typedef_union_u

Sample 15 typedef union with two members.

This is the detailed description for this typedef union.

Data Fields

- sample15_union_member1_params_s_type lb_cs_end
- sample15_union_member2_params_s_type lb_ps_end

4.2.2.15.1 Field Documentation

4.2.2.15.1.1 sample15_union_member1_params_s_type sample15_typedef_union_u::lb_cs_end

Sample 15 union member 1.

4.2.2.15.1.2 sample15_union_member2_params_s_type sample15_typedef_union_u::lb_ps_end

Sample 15 union member 2.

4.2.2.16 union sample16_union_with_struct_members_u

Sample 16 union with two struct members.

Data Fields

- struct {
  sample16_timer_id_T lb_timer_id
  } **sample16_struct_member1**

- struct {
  sample16_utimer_id_T lb_utimer_id
  } **sample16_struct_member2**

4.2.2.16.1 Field Documentation

4.2.2.16.1.1 sample16_timer_id_T sample16_union_with_struct_members_u::lb_timer_id

Sample 16 struct member 1.

4.2.2.16.1.2 sample16_utimer_id_T sample16_union_with_struct_members_u::lb_utimer_id

Sample 16 struct member 2.

4.2.2.17 struct sample17_typedef_struct_type

This a very complex sample typedef struct, which includes one enum type parameter, two union members and one struct member. Each union member includes two levels of nested struct members.

Data Fields

- sample17_enum_type_param np_vsn
- union {
  struct {
  npfltr_np4_hdr_field_mask_type field_mask
  npfltr_np4_hdr_field_mask_type err_mask
  struct {
  struct ps_in_addr addr
  struct ps_in_addr subnet_mask
  } nested_struct_member1_of_struct1_of_union_member1
  struct {
  struct ps_in_addr addr
  struct ps_in_addr subnet_mask
  } **nested_struct_member2_of_struct1_of_union_member1**
  struct {
  uint8 val
  uint8 mask

```
    } nested_struct_member3_of_struct1_of_union_member1
  } sample17_struct_member1_of_union_member1
  struct {
    npfltr_np6_hdr_field_mask_type field_mask
    npfltr_np6_hdr_field_mask_type err_mask
    struct {
      struct ps_in6_addr addr
      uint8 prefix_len
    } sample17_nested_struct_member1_of_struct2_of_union_member1
    struct {
      struct ps_in6_addr addr
      uint8 prefix_len
    } sample17_nested_struct_member2_of_struct2_of_union_member1
    struct {
      uint8 val
      uint8 mask
    } sample17_nested_struct_member3_of_struct2_of_union_member1
    uint32 flow_label
    uint8 next_hdr_prot
  } sample17_struct_member2_of_union_member1
} sample17_union_member1
```

- union {
  struct {
  npfltr_tcp_hdr_field_mask_type field_mask
  npfltr_tcp_hdr_field_mask_type err_mask
  struct {
  uint16 port
  uint16 range
  } sample17_nested_struct_member1_of_struct1_of_union_member2
  struct {
  uint16 port
  uint16 range
  } sample17_nested_struct_member2_of_struct1_of_union_member2
  } sample17_struct_member1_of_union_member2
  struct {
  npfltr_udp_hdr_field_mask_type field_mask
  npfltr_udp_hdr_field_mask_type err_mask
  struct {
  uint16 port
  uint16 range
  } sample17_nested_struct_member1_of_struct_member2_of_union_member2
  struct {
  uint16 port
  uint16 range
  } sample17_nested_struct_member2_of_struct_member2_of_union_member2
  } sample17_struct_member2_of_union_member2
  struct {
  npfltr_icmp_hdr_field_mask_type field_mask

npfltr_icmp_hdr_field_mask_type err_mask
uint8 type
uint8 code
} **sample17_struct_member3_of_union_member2**
struct {
npfltr_esp_hdr_field_mask_type field_mask
npfltr_esp_hdr_field_mask_type err_mask
uint32 spi
} **sample17_struct_member4_of_union_member2**
struct {
npfltr_tcp_udp_hdr_field_mask_type field_mask
npfltr_tcp_udp_hdr_field_mask_type err_mask
struct {
uint16 port
uint16 range
} **sample17_nested_struct_member1_of_struct_member5_of_union_member2**
struct {
uint16 port
uint16 range
} **sample17_nested_struct_member2_of_struct_member5_of_union_member2**
} **sample17_struct_member5_of_union_member2**
} **sample17_union_member2**

- struct {
  uint16 fi_id
  uint16 fi_precedence
  } **sample17_struct_member1_of_typedef_struct_type**

4.2.2.17.1 Field Documentation

4.2.2.17.1.1 sample17_enum_type_param sample17_typedef_struct_type::np_vsn

Sample 17 enum type parameter.

4.2.2.17.1.2 npfltr_np4_hdr_field_mask_type sample17_typedef_struct_type::field_mask

In mask.

4.2.2.17.1.3 npfltr_np4_hdr_field_mask_type sample17_typedef_struct_type::err_mask

Out mask.

4.2.2.17.1.4 struct ps_in_addr sample17_typedef_struct_type::addr

Sample 17 subnested struct member 1 of struct member 1 of union member 1.

Sample 17 subnested struct member 1 of struct member 2 of union member 1.

4.2.2.17.1.5 struct ps_in_addr sample17_typedef_struct_type::subnet_mask

Sample 17 subnested struct member 2 of struct member 2 of union member 1.

4.2.2.17.1.6 struct { ... } ::@15 sample17_typedef_struct_type::nested_struct_member1_of_struct1_of_union_member1

Comment for nested struct 1 shows up, but adds an odd character 17@ in the heading.

4.2.2.17.1.7 uint8 sample17_typedef_struct_type::val

A value.

Traffic class value.

4.2.2.17.1.8 uint8 sample17_typedef_struct_type::mask

A mask.

Traffic class mask.

4.2.2.17.1.9 struct { ... } sample17_typedef_struct_type::sample17_struct_member1_of_union_member1

Comment shows up if I put it here.

4.2.2.17.1.10 npfltr_np6_hdr_field_mask_type sample17_typedef_struct_type::field_mask

In mask.

4.2.2.17.1.11 npfltr_np6_hdr_field_mask_type sample17_typedef_struct_type::err_mask

Out mask.

4.2.2.17.1.12 uint8 sample17_typedef_struct_type::prefix_len

Length of the prefix.

4.2.2.17.1.13 uint32 sample17_typedef_struct_type::flow_label

Flow label.

4.2.2.17.1.14 uint8 sample17_typedef_struct_type::next_hdr_prot

Transport-level protocol header.

4.2.2.17.1.15 npfltr_tcp_hdr_field_mask_type sample17_typedef_struct_type::field_mask

In mask.

4.2.2.17.1.16 npfltr_tcp_hdr_field_mask_type sample17_typedef_struct_type::err_mask

Out mask.

4.2.2.17.1.17 uint16 sample17_typedef_struct_type::port

NP2 source port.

NP3 destination port.

NP4 source port.

NP4 destination port.

Source port.

Destination port.

4.2.2.17.1.18 uint16 sample17_typedef_struct_type::range

Range of the NP2 source port.

Range of the NP3 destination port.

Range of the NP4 source port.

Range of the UDP destination port.

Range of the source port.

Range of the destination port.

4.2.2.17.1.19 npfltr_udp_hdr_field_mask_type sample17_typedef_struct_type::field_mask

In mask.

4.2.2.17.1.20 npfltr_udp_hdr_field_mask_type sample17_typedef_struct_type::err_mask

Out mask.

4.2.2.17.1.21 npfltr_icmp_hdr_field_mask_type sample17_typedef_struct_type::field_mask

In mask.

4.2.2.17.1.22 npfltr_icmp_hdr_field_mask_type sample17_typedef_struct_type::err_mask

Out mask.

4.2.2.17.1.23 uint8 sample17_typedef_struct_type::type

NP5 type.

4.2.2.17.1.24 uint8 sample17_typedef_struct_type::code

NP5 code.

4.2.2.17.1.25 npfltr_esp_hdr_field_mask_type sample17_typedef_struct_type::field_mask

In mask.

4.2.2.17.1.26 npfltr_esp_hdr_field_mask_type sample17_typedef_struct_type::err_mask

Out mask.

4.2.2.17.1.27 uint32 sample17_typedef_struct_type::spi

Secure index.

4.2.2.17.1.28 npfltr_tcp_udp_hdr_field_mask_type sample17_typedef_struct_type::field_mask

In mask.

4.2.2.17.1.29 npfltr_tcp_udp_hdr_field_mask_type sample17_typedef_struct_type::err_mask

Out mask.

4.2.2.17.1.30 uint16 sample17_typedef_struct_type::fi_id

Filter ID.

4.2.2.17.1.31 uint16 sample17_typedef_struct_type::fi_precedence

Filter precedence.

4.2.2.18 struct oob_area_info

Keeps track of out-of-bounds block area information and the size of each block.

Data Fields

- uint8 block_count
- uint8 block_size_in_bytes

4.2.2.18.1 Field Documentation

4.2.2.18.1.1 uint8 oob_area_info::block_count

Number of blocks in the out-of-bounds area.

4.2.2.18.1.2 uint8 oob_area_info::block_size_in_bytes

Size of each block in the out-of-bounds area.

4.2.2.19 struct local_area

Data Fields

- uint32 area_count
- uint32 area_size_in_bytes

4.2.2.19.1 Field Documentation

4.2.2.19.1.1 uint32 local_area::area_count

Total number of blocks in the local area.

4.2.2.19.1.2 uint32 local_area::area_size_in_bytes

Size of each block in the local area.

4.2.2.20 struct sample2_parent_struct_with_members_child_struct_as_member::child_struct_member

Data Fields

- int x1
- int y1

4.2.2.20.1 Field Documentation

4.2.2.20.1.1 int sample2_parent_struct_with_members_child_struct_as_member::child_struct_member::x1

Integer x1 in child struct.

4.2.2.20.1.2 int sample2_parent_struct_with_members_child_struct_as_member::child_struct_member::y1

Integer y1 in child struct.

4.2.3 Define Documentation

4.2.3.1 #define FANCY_COLD_BOOT_START_BLOCK_VALUE

Initial value for the start block.

4.2.3.2 #define FANCY_MAX_COLD_BOOT_END_BLOCK_VALUE

Maximum value for the end block.

4.2.3.3 #define FANCY_COLD_BOOT_BLOCK_VALUE_INVALID

Value for the cold boot block is invalid.

4.2.3.4 #define FANCY_COLD_BOOT_BLOCK_VALUE_UNASSIGNED

Value for the cold boot block is unassigned.

4.2.4 Typedef Documentation

4.2.4.1 typedef struct _sample11_struct_sample_gtx_t sample11_typedefstruct_sample_gtx_t

Sample 11 typedef struct with a struct member that has an enum and a union member. Note that the comment for struct member 3 (enum) and struct member 4 (union) do not show up in PDF under this typedef struct as it should. In fact, they don't show up anywhere in the PDF. Dimitri fixing as part of SOW task 2f.

4.3 Second Fancy Device API

This is a brief description for the Second Fancy Device API.

Data Structures

- class CRectangle
- struct fancy_warm_boot_block

Namespaces

- namespace AR

  Access rights namespace.
- namespace fancy_audio_1

  Provides SamplerState and ChannelMode1 enumeration types.

Defines

- #define FANCY_WARM_BOOT_START_BLOCK_VALUE
- #define FANCY_MAX_WARM_BOOT_END_BLOCK_VALUE
- #define FANCY_WARM_BOOT_BLOCK_VALUE_INVALID
- #define FANCY_WARM_BOOT_BLOCK_VALUE_UNASSIGNED_WITHAVERYLONGLONGDEFINENAMEWITHNOUNDERSCORES

Enumerations

- enum download_type { download_OOB, download_NOOB, download_ONENOOB, download_MULTI }
- enum download_tech { download_SLC, download_MLC }
- enum download_ID { download_x8, download_x16 }
- enum { **FANCY_READ_MAIN**, **FANCY_READ_SPARE**, **FANCY_READ_MAIN_SPARE**, **FANCY_READ_RAW**, **FANCY_READ_BYTES** }

Functions

- void **CRectangle::set_values** (int, int)

4.3.1 Detailed Description

This is the detailed description for the second API group.

4.3.2 Data Structure Documentation

4.3.2.1 class CRectangle

Takes the height and length of a rectangle and returns the area value.

This class (CRectangle) contains four members: two input parameter members of type integer (x, y, where x=height and y=length) and two member functions (set_values, area) with public access.

This class calls the set_values member function to set the input values (x, y) to integers. This function is void, which indicates that is does not return a result; it just holds the results temporarily. It then calls the area member function, which takes the values in the set_values member function (x, y), multiplies them (x∗y), and returns the result (area of the rectangle).

Parameters

in	*x*	Data member x of type integer with private access. This member is private by default. This sentence was added to test out the single-spacing issue for long descriptions inside a parameter table.
in	*y*	Data member y of type integer with private access. This member is private by default.

Returns

Area of the rectangle as an integer value.

Dependencies

None.

Public Member Functions

- void **set_values** (int, int)
- int **area** ()

Data Fields

- int **x**
- int **y**

4.3.2.2 struct fancy_warm_boot_block

Holds the new and previous warm boot block numbers.

This is an optional detailed description. This structure is used in Fancy Device tools during software download.

Data Fields

- int warm_boot_new_block
- int warm_boot_old_block

4.3.2.2.1 Field Documentation

4.3.2.2.1.1 int fancy_warm_boot_block::warm_boot_new_block

New warm boot block number.

4.3.2.2.1.2 int fancy_warm_boot_block::warm_boot_old_block

Old warm boot block number.

4.3.3 Define Documentation

4.3.3.1 #define FANCY_WARM_BOOT_START_BLOCK_VALUE

Initial value for the start block.

4.3.3.2 #define FANCY_MAX_WARM_BOOT_END_BLOCK_VALUE

Maximum value for the end block.

4.3.3.3 #define FANCY_WARM_BOOT_BLOCK_VALUE_INVALID

Warm boot block value is invalid.

4.3.3.4 #define FANCY_WARM_BOOT_BLOCK_VALUE_UNASSIGNED_WITHAVERYLONGLONGDEFINENAMEWITHNOUNDERSCORES

Warm boot block value is unassigned.

4.3.4 Enumeration Type Documentation

4.3.4.1 enum download_type

Identifies the device types for downloading software to one or more devices.

Enumerator:

download_OOB Out-Of-Bounds device.
download_NOOB More than one Not-Out-Of-Bounds device.
Note: This is an example of a note for an enum member (note1 command). This can also be used in param descriptions.
download_ONENOOB OneNOOB device.
download_MULTI Multiple devices.

4.3.4.2 enum download_tech

Identifies the bits per cell for the device.

Enumerator:

download_SLC Single-Level Cell device.
download_MLC Multi-Level Cell device.

4.3.4.3 enum download_ID

Identifies the download ID.

Enumerator:

download_x8 8-bit interface download ID.
download_x16 16-bit interface download ID.

4.4 Third Fancy Device API

This is a brief description for the third Fancy Device API.

Data Structures

- struct fancy_cold_boot_block
- union _AfancyAudioEvent
- struct PACKED_POST
- struct FancyIAO

Namespaces

- namespace fancy_audio_2

  Provides ResamplerState and ChannelMode2 enumeration types.

Modules

- Subgroup of Third Fancy Device API

  This is a brief description for the subgroup of the third Fancy Device API.

Defines

- #define FANCY_DEVICE_DONE
- #define FANCY_DEVICE_FAIL
- #define FANCY_DEVICE_NOT_SUPPORTED
- #define FANCY_DEVICE_CP_READ_FAIL

Typedefs

- typedef PACKED struct PACKED_POST **fancy_qos_params_type**
- typedef PACKED enum PACKED_POST **fancy_header_comp_e_type**

Enumerations

- enum PACKED_POST { FANCY_HEADER_COMP_OFF, FANCY_HEADER_COMP_ON, FANCY_HEADER_COMP_MAX }
- enum fancy_dev_val_e_type { FANCY_DEVICE_VAL_OFF, FANCY_DEVICE_VAL_ON }
- enum FancySysTimeType1 { **IAO_SYSTIME_UMTS**, IAO_SYSTIME_GSM, IAO_SYSTIME_TOT }
- enum WfdWSEvtType { WFD_WS_EVENT_TOT }

Functions

- fancy_return_type fancy_report_event (const fancy_event_data_type *data)
- static int CreateInstance (FANCYCLID clsid, IEnv *pEnvironment, IPrivSet *pPrivSet, void **ppNewObj)
- virtual int NDQueryInterface (FANCYID idReq, IQI **ppIface)
- FancyResult NameThread ()
- FancyInterfere (FANCYCLSID clsid, pEnv *pEnvironment, int &result)
- int printf (const char *fmt,...)

4.4.1 Detailed Description

This is the detailed description for the third API group.

4.4.2 Data Structure Documentation

4.4.2.1 struct fancy_cold_boot_block

Holds the new and previous cold boot block numbers.

This is an optional detailed description. This structure is used in Fancy Device tools during software initialization.

Data Fields

- int cold_boot_new_block
- int cold_boot_old_block

4.4.2.1.1 Field Documentation

4.4.2.1.1.1 int fancy_cold_boot_block::cold_boot_new_block

New cold boot block number.

4.4.2.1.1.2 int fancy_cold_boot_block::cold_boot_old_block

Old cold boot block number.

4.4.2.2 union _AfancyAudioEvent

Accesses the header values. This union includes two data structures.

Data Fields

- struct AfancyAudioAnyEvent **any**
- struct AfancyAudioAnyEvent **header**

4.4.2.3 struct PACKED_POST

Stores Fancy Quality of Service parameters.

This is an optional detailed description. This structure includes three members.

Data Fields

- boolean valid_flg
- uint32 precedence
- uint32 mean

4.4.2.3.1 Field Documentation

4.4.2.3.1.1 boolean PACKED_POST::valid_flg

Indicates whether the parameters are set and valid. This is a test detailed sentence.

4.4.2.3.1.2 uint32 PACKED_POST::precedence

Precedence class. This is a test detailed sentence.

4.4.2.3.1.3 uint32 PACKED_POST::mean

Mean throughput class.

4.4.2.4 struct FancyIAO

Fancy input and output device handles.

These handles allow the device to be configured for opening, closing, and synchronizing.

Data Fields

- FancyDevice **FancyDevice**
- FancyResult(∗ FancyOpen)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum)
- FancyResult(∗ FancyClose)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum)
- FancyResult(∗ FancyAsyncRead)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum, FANCYSYSMemHandle hXferBufMem, uint32 fancyLen, FANCYSYSEventHandle hEvent)
- FancyResult(∗ FancyAsyncWrite)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum, FANCYSYSMemHandle hXferBufMem, uint32 fancyLen, FANCYSYSEventHandle hEvent)

4.4.2.4.1 Field Documentation

4.4.2.4.1.1 FancyResult(∗ FancyIAO::FancyOpen)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum)

Main fancy device.

4.4.2.4.1.2 FancyResult(∗ FancyIAO::FancyClose)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum)

Open fancy device.

4.4.2.4.1.3 FancyResult(∗ FancyIAO::FancyAsyncRead)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum, FANCYSYSMemHandle hXferBufMem, uint32 fancyLen, FANCYSYSEventHandle hEvent)

Close fancy device.

4.4.2.4.1.4 FancyResult(∗ FancyIAO::FancyAsyncWrite)(FancyDeviceHandle ∗_h, uint32 fancyFifoNum, FANCYSYSMemHandle hXferBufMem, uint32 fancyLen, FANCYSYSEventHandle hEvent)

Asynchronous read of fancy device.

4.4.3 Define Documentation

4.4.3.1 #define FANCY_DEVICE_DONE

Operation passed.

4.4.3.2 #define FANCY_DEVICE_FAIL

Operation failed.

4.4.3.3 #define FANCY_DEVICE_NOT_SUPPORTED

Device is not supported.

4.4.3.4 #define FANCY_DEVICE_CP_READ_FAIL

Copy page read failure.

4.4.4 Enumeration Type Documentation

4.4.4.1 enum PACKED_POST

Fancy header compression types.

Enumerator:

FANCY_HEADER_COMP_OFF Compression is off (default).
FANCY_HEADER_COMP_ON Compression is on when selected.
FANCY_HEADER_COMP_MAX Forces the maximum compression to 0xff so the enumeration is defined as a byte.

4.4.4.2 enum fancy_dev_val_e_type

Fancy device validation modes.

Enumerator:

FANCY_DEVICE_VAL_OFF Device validation mode is off (default).
FANCY_DEVICE_VAL_ON Device validation mode is on when selected.

4.4.4.3 enum FancySysTimeType1

Enumerator:

IAO_SYSTIME_GSM GSM system time.
IAO_SYSTIME_TOT Total number of system time types.

4.4.4.4 enum WfdWSEvtType

Supported WFD warm start events.

Enumerator:

WFD_WS_EVENT_TOT Total number of warm start events

4.4.5 Function Documentation

4.4.5.1 fancy_return_type fancy_report_event (data *data*)

Event-driven drivers call this function to report asynchronous events to the Fancy Device.

Parameters

in	*data*	Event-related data.

Returns

FANCY_SUCCESS – Requested operation was successful.
FANCY_FAILURE – Requested operation was not successful.
FANCY_LOCKED – Operation failed because device was locked.

Dependencies

None.

Examples:

fancy_report_event_example.c.

4.4.5.2 static int CreateInstance (clsid, *clsid,* pEnvironment, *pEnvironment,* pPrivSet, *pPrivSet,* ppNewObj *ppNewObj*) [static]

Provides a public entry point for the module. A new instance of FancyInterfere is created for FancyModule, and the default interface is returned.

Message payload

The following table is an example of a table which was created using the Word-to-LaTeX tool with minor manual formatting adjustments:

Type	Parameter	Supported values	Description

uint32	uProfilingLevel	0 – Turn off profiling 1 – Collect summary information of MIPS/memory 2 – Collect summary information of MIPS/memory, per-software thread MIPS, and stack consumption	
uint32	uPhyAddress	Physical address where the aDSP can write the profiling data for each event	Must be 4 K-aligned.
uint32	uSize	Amount of available space in bytes for writing profiling data	Must be a multiple of 4 K page size
uint32	uSamplingPeriod	$\geq$ 100000	Number of μs between successive events

Note

Test note 1.
Test note 2.
Test note 3.

Parameters

`in`	*clsid*	Class ID of the module.
`in`	*pEnvironment*	Interface to the Fancy Services environment.
`out`	*pPrivSet*	Privilege set of the caller.
`in,out`	*ppNewObj*	Set to the interface pointer of the new instance.

Returns

SUCCESS – Instance was created successfully.
ENOMEMORY – Indicates a memory allocation failure.

Dependencies

None.

Side effects

An invalid class ID may produce erroneous results.

See also

NDQueryInterface

4.4.5.3 virtual int NDQueryInterface (idReq, *idReq,* ppIface *ppIface*) `[virtual]`

Handles interface pointers for non-based classes.

This method must be implemented if the FancyInterfere module supports any interfaces not handled by its base classes.

Note

Test note 1.
Test note 2.
Test note 3.
Test note 4.
Test note 5.
Test note 6.
Test note 7.
Test note 8.
Test note 9.
Test note 10.

Parameters

in	*idReq*	Unique ID of the requested interface.
out	*ppIface*	Interface pointer of the requested interface.

Returns

SUCCESS – Interface is returned successfully.
ECLASSNOTSUPPORT – Interface is not supported.

4.4.5.4 FancyResult NameThread ( )

Handles thread name intialization.

The deriving class can override this method to provide a descriptive name for the optional output thread. This helps distinguish it in debugging applications. The default implementation provides a generic name.

Note: This is an example of a single hanging indent note (note1hang command). It is primarily intended for use in fancy tools during software download.

Returns

SUCCESS – Initialization was successful.
FAILURE – Initialization failed.

4.4.5.5 FancyInterfere (clsid, *clsid,* pEnvironment, *pEnvironment,* result *result*)

Provides a class constructor.

This is the component constructor, meant to be called by the class factory of the component.

Parameters

in	*clsid*	Class ID of the module.
in	*pEnvironment*	Interface to the Fancy Services environment.
out	*result*	Result of the function.

Returns

The following results may be returned:
- Valid class object

- Invalid class object

4.4.5.6 int printf (fmt, *fmt,* ... ...)

Standard print function. The extern keyword is used to inform the compiler about variables declared outside of the current file. Variables described by extern statements do not have any space allocated for them because they have been previously defined elsewhere.

4.5 Subgroup of Third Fancy Device API

This is a brief description for the subgroup of the third Fancy Device API.

This is the detailed description for the subgroup of the third API group.

4.6 Fancy Device Common Data

This is a brief description for the Fancy Device Common Data group.

- typedef struct fancy_handle **fancy_handle**
- typedef fancy_handle ∗ **fancy_handle_t**
- #define FANCY_DEFAULT_DATA_PROFILE_VERSION

4.6.1 Detailed Description

This is the detailed description for the Fancy Device Common Data group.

4.6.2 Define Documentation

4.6.2.1 #define FANCY_DEFAULT_DATA_PROFILE_VERSION

Default profile version. The value of the Fancy Device target's profile version depends on the following conditional settings:

- If FANCY_DATA_TARGET1 is TRUE, FANCY_DATA_PROFILE_VERSION = 7.
- If FANCY_DATA_TARGET2 is TRUE, FANCY_DATA_PROFILE_VERSION = 9.
- If FANCY_DATA_TARGET3 is TRUE, FANCY_DATA_PROFILE_VERSION = 12.
- If none of the above conditions are TRUE, FANCY_DEFAULT_DATA_PROFILE_VERSION = 3.

4.7 Deprecated_example

Functions

- test_enum_type test_process_nice_list (item_type ∗∗nice_list_msg_ptr, uint16 msg_len, nice_list_pdata_type ∗sys_nice_list, uint32 ∗security_alert_mask_ptr, boolean ignore_expiration)
- test_enum_type test_process_bad_list (item_type ∗∗bad_list_msg_ptr, uint16 msg_len, bad_list_pdata_type ∗sys_bad_list, uint32 ∗security_alert_mask_ptr, boolean ignore_expiration)

4.7.1 Function Documentation

4.7.1.1 test_enum_type test_process_nice_list (nice_list_msg_ptr, *nice_list_msg_ptr,* msg_len, *msg_len,* sys_nice_list, *sys_nice_list,* security_alert_mask_ptr, *security_alert_mask_ptr,* ignore_expiration *ignore_expiration*)

Deprecated

Use the new interface to process a nice list.

This function processes a nice list, by parsing the nice list from the server up to the top.

Parameters

in	*nice_list_msg_ptr*	Pointer to the message containing the nice list.
out	*longkey*	Long public key.
out	*sys_nice_list*	System nice list.
in	*security_alert_mask_ptr*	Alert mask for processing errors.
in	*ignore_expiration*	If the time validity of the list is to be checked. TRUE – Do not check. FALSE – Check.

Returns

E_SUCCESS – Processing is successful.
E_DATA_INVALID – List is not valid.
E_DATA_TOO_LARGE – List is too large for the buffer.
E_NICELIST_FAILURE – Failed to process the nice list due to other reasons.

4.7.1.2 test_enum_type test_process_bad_list (bad_list_msg_ptr, *bad_list_msg_ptr,* msg_len, *msg_len,* sys_bad_list, *sys_bad_list,* security_alert_mask_ptr, *security_alert_mask_ptr,* ignore_expiration *ignore_expiration*)

Deprecated

Use the new interface to process a bad list.

This function processes a bad list, by parsing the bad list from the server up to the top.

Parameters

in	*bad_list_msg_ptr*	Pointer to the message containing the bad list.
out	*longkey*	Long public key.
out	*sys_bad_list*	System bad list.
in	*security_alert_mask_ptr*	Alert mask for processing errors.
in	*ignore_expiration*	If the time validity of the list is to be checked. TRUE – Do not check. FALSE – Check.

Returns

E_SUCCESS – Processing is successful.
E_DATA_INVALID – List is not valid.
E_DATA_TOO_LARGE – List is too large for the buffer.
E_BADLIST_FAILURE – Failed to process the bad list due to other reasons.

5 Namespace Documentation

5.1 AR Namespace Reference

Access rights namespace.

Enumerations

- enum **RightsChangeReason** { **RightsChangeReason_Added**, **RightsChangeReason_Deleted**, **RightsChangeReason_Modified**, **RightsChangeReason_Unknown**, **_AR_PLACEHOLDER_RightsChangeReason** }

Variables

- const ::FANCYID FANCYID_IRightsChange

5.1.1 Detailed Description

Detailed description for the AR namespace.

5.1.2 Variable Documentation

5.1.2.1 const ::FANCYID AR::FANCYID_IRightsChange

Changes the access rights.

5.2 fancy_audio_1 Namespace Reference

Provides SamplerState and ChannelMode1 enumeration types.

Enumerations

- enum **ResamplerState** { **RS_OK**, **RS_NEED_INPUT_BUF**, **RS_NEED_OUTPUT_BUF** }
- enum **ChannelMode1** { **MONO**, **STEREO** }

Functions

- static int CreateInstance (FANCYCLSID clsid, IEnv *pEnvironment, IPrivSet *pPrivSet, void **pp-NewObj)

Variables

- GenericResamplerLib * m_pResampler
- bool m_Enabled
- ICritSect * m_pRxBufferLock
- std::queue< FancyCommand * > m_fancyRxBufferQueue
- std::queue< FancyCommand * > **m_fancyTxBufferQueue**
- DECLARE_IQI

5.2.1 Detailed Description

This is the first fancy audio namespace.

5.2.2 Function Documentation

5.2.2.1 static int fancy_audio_1::CreateInstance (clsid, *clsid,* pEnvironment, *pEnvironment,* pPrivSet, *pPrivSet,* ppNewObj *ppNewObj*) [static]

Provides a public entry point for the module.

A new instance of FancyPlayback is created, and the default interface (FancyMediaModule) is returned.

Parameters

in	*clsid*	FANCYCLSID of the module.
in	*pEnvironment*	Interface to the Fancy Services environment.
out	*pPrivSet*	Privilege set of the caller.
in,out	*ppNewObj*	Interface pointer of the new instance.

Returns

SUCCESS – Instance was created successfully.
ENOMEMORY – Memory allocation failure.

Comments

A memory allocation failure requires a reboot.

Errors

If the new environment value is not an integer, the compiler generates an error.
If the new environment object is called outside the Fancy Services environment without the pointer, a constraint error occurs.

5.2.3 Variable Documentation

5.2.3.1 GenericResamplerLib* fancy_audio_1::m_pResampler

Generic Resampler library object.

5.2.3.2 bool fancy_audio_1::m_Enabled

Enables the master flag during FancyPlayback.

5.2.3.3 ICritSect* fancy_audio_1::m_pRxBufferLock

Critical section for the Rx buffer queue.

5.2.3.4 std::queue<FancyCommand*> fancy_audio_1::m_fancyRxBufferQueue

Rx and Tx queues for Fancy Device buffers received from the application.

5.2.3.5 fancy_audio_1::DECLARE_IQI

Pulls in IQueryInterface method definitions.

5.3 fancy_audio_2 Namespace Reference

Provides ResamplerState and ChannelMode2 enumeration types.

Enumerations

- enum SamplerState { S_OK, S_NEED_INPUT_BUF, S_NEED_OUTPUT_BUF }
- enum ChannelMode2 { MONO, STEREO }

Functions

- virtual int CDECL NDQueryInterface (FANCYCLSIDRQ idReq, IQI **ppIface)
- FancyResult NameThread ()
- FancyInterfere (FANCYCLSID clsid, IEnv *pEnvironment, int &result)

Variables

- GenericSamplerLib * m_pResampler
- uint32 m_uiInpBufferSize
- boolean m_fSampTypeSet
- bool m_Enabled

- bool m_EnabledRead
- ISignalQ * m_pEventSignalQ
- std::queue< FancyCommand * > m_fancyRxBufferQueue
- std::queue< FancyCommand * > m_fancyTxBufferQueue
- DECLARE_IQI

5.3.1 Detailed Description

This is the detailed fancy audio namespace.

5.3.2 Enumeration Type Documentation

5.3.2.1 enum fancy_audio_2::SamplerState

Sample comment for enum inside a namespace.

Enumerator:

S_OK Resampler state is in OK mode.
S_NEED_INPUT_BUF Resampler state needs an input buffer.
S_NEED_OUTPUT_BUF Resampler state needs an output buffer.

5.3.2.2 enum fancy_audio_2::ChannelMode2

Another sample comment for enum inside a namespace.

Enumerator:

MONO Channel Mode 2 is in mono mode.
STEREO Channel Mode 2 is in stereo mode.

5.3.3 Function Documentation

5.3.3.1 virtual int CDECL fancy_audio_2::NDQueryInterface (idReq, *idReq,* ppIface *ppIface*) [virtual]

Provides an interface to support the FancyInterface module.

This method must be implemented if the FancyInterfere module supports any interfaces not handled by its base classes.

Parameters

in	*idReq*	Unique ID of the requested interface.
out	*ppIface*	Interface pointer of the requested interface.

Returns

SUCCESS – Interface is returned successfully.
ECLASSNOTSUPPORT – Interface is not supported.

5.3.3.2 FancyResult fancy_audio_2::NameThread ( )

Overrides a generic name.

The deriving class may override this method to provide a descriptive name for the optional output thread to help distinguish it in debugging applications. The default implementation provides a generic name.

Returns

SUCCESS – Initialization was successful.
FAILURE – Initialization failed.

5.3.3.3 fancy_audio_2::FancyInterfere (clsid, *clsid,* pEnvironment, *pEnvironment,* result *result*)

Provides a class constructor.

This is the component constructor, meant to be called by the class factory of the component.

Parameters

in	*clsid*	Class ID of the module.
in	*pEnvironment*	Interface to the component services environment.
out	*result*	Result of the function.

Returns

New class object.

5.3.4 Variable Documentation

5.3.4.1 GenericSamplerLib* fancy_audio_2::m_pResampler

Generic Sampler library object.

5.3.4.2 uint32 fancy_audio_2::m_uiInpBufferSize

Size of the input buffer delivered to the sampler.

5.3.4.3 boolean fancy_audio_2::m_fSampTypeSet

SamplerType Set flag.

5.3.4.4 bool fancy_audio_2::m_Enabled

Enables the flag master for FancyInterfere.

5.3.4.5 bool fancy_audio_2::m_EnabledRead

Enables FancyInterfere on the Read channel.

5.3.4.6 ISignalQ* fancy_audio_2::m_pEventSignalQ

Signal queue for blocking events.

5.3.4.7 std::queue<FancyCommand*> fancy_audio_2::m_fancyRxBufferQueue

Rx queues for Fancy Device buffers received from the application.

5.3.4.8 std::queue<FancyCommand*> fancy_audio_2::m_fancyTxBufferQueue

Tx queues for Fancy Device buffers received from the application.

5.3.4.9 fancy_audio_2::DECLARE_IQI

Pulls in IQueryInterface method definitions. Data Structure Documentation

6 Example Documentation

6.1 fancy_report_event_example.c

This is an example of how the example feature works.

```
/* Reports an event concerning fancy device detection. */

#define FANCY_EVENT_DATA_VER  2

fancy_event_data_type   fancy_data;
fancy_src_state_type    state = FANCY_SRC_STATE_HI;

fancy_data.ve                   = (uint32) FANCY_EVENT_DATA_VER;
fancy_data.id                   = FANCY_EVNT_DEV;
fancy_data.length               = sizeof (fancy_src_state_type);
fancy_data.data.hsd_event_data  = (fancy_src_state_type*) &state;

fancy_report_event (&hsd_data);
```

Index