## RDC

Generated by Doxygen 1.8.11

# **Contents**

1	Data	Struct	ture Index	1
	1.1	Data S	Structures	 1
2	File	Index		3
	2.1	File Lis	ist	 3
3	Data	Struct	ture Documentation	5
	3.1	rdc_de	evice_attributes_t Struct Reference	 5
		3.1.1	Detailed Description	 5
	3.2	rdc_fie	eld_group_info_t Struct Reference	 5
		3.2.1	Detailed Description	 6
		3.2.2	Field Documentation	 6
			3.2.2.1 field_ids	 6
	3.3	rdc_fie	eld_value Struct Reference	 6
		3.3.1	Detailed Description	 6
		3.3.2	Field Documentation	 7
			3.3.2.1 value	 7
	3.4	rdc_gp	pu_usage_info_t Struct Reference	 7
		3.4.1	Detailed Description	 8
	3.5	rdc_gr	roup_info_t Struct Reference	 8
		3.5.1	Detailed Description	 8
		3.5.2	Field Documentation	 8
			3.5.2.1 entity_ids	 8
	3.6	rdc_jol	bb_group_info_t Struct Reference	 8
		3.6.1	Detailed Description	 9
	3.7	rdc_jol	bb_info_t Struct Reference	 9
		3.7.1	Detailed Description	9
		3.7.2	Field Documentation	 9
			3.7.2.1 summary	 9
	3.8	rdc_sta	tats_summary_t Struct Reference	10
			Detailed Description	10

iv CONTENTS

4	File	Docume	umentation 11		11
	4.1	rdc.h F	ile Referer	nce	11
		4.1.1	Detailed	Description	14
		4.1.2	Macro De	efinition Documentation	15
			4.1.2.1	RDC_FI_GPU_MEMORY_USAGE	15
			4.1.2.2	RDC_FI_GPU_MEMORY_TOTAL	15
			4.1.2.3	RDC_FI_POWER_USAGE	15
			4.1.2.4	RDC_FI_GPU_CLOCK	15
			4.1.2.5	RDC_FI_MEM_CLOCK	15
			4.1.2.6	RDC_FI_PCIE_TX	15
			4.1.2.7	RDC_FI_PCIE_RX	15
			4.1.2.8	RDC_FI_GPU_UTIL	15
			4.1.2.9	RDC_FI_ECC_CORRECT_TOTAL	15
			4.1.2.10	RDC_FI_ECC_UNCORRECT_TOTAL	15
			4.1.2.11	RDC_FI_MEMORY_TEMP	16
			4.1.2.12	RDC_FI_GPU_TEMP	16
			4.1.2.13	RDC_FI_GPU_COUNT	16
			4.1.2.14	RDC_FI_DEV_NAME	16
		4.1.3	Typedef [	Documentation	16
			4.1.3.1	rdc_handle_t	16
		4.1.4	Enumera	tion Type Documentation	16
			4.1.4.1	rdc_status_t	16
			4.1.4.2	rdc_group_type_t	17
		4.1.5	Function	Documentation	17
			4.1.5.1	rdc_init(uint64_t init_flags)	17
			4.1.5.2	rdc_shutdown()	17
			4.1.5.3	rdc_start_embedded(rdc_operation_mode_t op_mode, rdc_handle_t *p_rdc_← handle)	17
			4.1.5.4	rdc_stop_embedded(rdc_handle_t p_rdc_handle)	18
			4.1.5.5	rdc_connect(const_char *ipAndPort, rdc_handle_t *p_rdc_handle, const_char *root_ca, const_char *client_cert, const_char *client_key)	18

CONTENTS

4.1.5.6	rdc_disconnect(rdc_handle_t p_rdc_handle)	18
4.1.5.7	rdc_job_start_stats(rdc_handle_t p_rdc_handle, rdc_gpu_group_t group_id, const char job_id[64], uint64_t update_freq)	20
4.1.5.8	rdc_job_get_stats(rdc_handle_t p_rdc_handle, const char job_id[64], rdc_job_⇔ info_t *p_job_info)	20
4.1.5.9	rdc_job_stop_stats(rdc_handle_t p_rdc_handle, const char job_id[64])	21
4.1.5.10	rdc_job_remove(rdc_handle_t p_rdc_handle, const char job_id[64])	21
4.1.5.11	rdc_job_remove_all(rdc_handle_t p_rdc_handle)	21
4.1.5.12	rdc_field_update_all(rdc_handle_t p_rdc_handle, uint32_t wait_for_update)	22
4.1.5.13	rdc_device_get_all(rdc_handle_t p_rdc_handle, uint32_t gpu_index_list[RDC_← MAX_NUM_DEVICES], uint32_t *count)	22
4.1.5.14	rdc_device_get_attributes(rdc_handle_t p_rdc_handle, uint32_t gpu_index, rdc⇔_device_attributes_t *p_rdc_attr)	22
4.1.5.15	rdc_group_gpu_create(rdc_handle_t p_rdc_handle, rdc_group_type_t type, const char *group_name, rdc_gpu_group_t *p_rdc_group_id)	23
4.1.5.16	rdc_group_gpu_add(rdc_handle_t p_rdc_handle, rdc_gpu_group_t group_id, uint32_t gpu_index)	23
4.1.5.17	rdc_group_gpu_get_info(rdc_handle_t p_rdc_handle, rdc_gpu_group_t p_rdc_⇔ group_id, rdc_group_info_t *p_rdc_group_info)	24
4.1.5.18	rdc_group_get_all_ids(rdc_handle_t p_rdc_handle, rdc_gpu_group_t group_id↔ _list[], uint32_t *count)	24
4.1.5.19	rdc_group_gpu_destroy(rdc_handle_t p_rdc_handle, rdc_gpu_group_t p_rdc_⇔ group_id)	24
4.1.5.20	$\label{lem:constraint} $$ rdc\_group\_field\_create(rdc\_handle\_t p\_rdc\_handle, uint32\_t num\_field\_ids, uint32\_t *field\_ids, const char *field\_group\_name, rdc\_field\_grp\_t *rdc\_field\_compunation="color: blue;" and the particle of the particl$	25
4.1.5.21	rdc_group_field_get_info(rdc_handle_t p_rdc_handle, rdc_field_grp_t rdc_field _group_id, rdc_field_group_info_t *field_group_info)	25
4.1.5.22	rdc_group_field_get_all_ids(rdc_handle_t p_rdc_handle, rdc_field_grp_t field_⇔ group_id_list[], uint32_t *count)	25
4.1.5.23	rdc_group_field_destroy(rdc_handle_t p_rdc_handle, rdc_field_grp_t rdc_field↔ _group_id)	26
4.1.5.24	rdc_field_watch(rdc_handle_t p_rdc_handle, rdc_gpu_group_t group_id, rdc_← field_grp_t field_group_id, uint64_t update_freq, double max_keep_age, uint32← t max_keep_samples)	26
4.1.5.25	rdc_field_get_latest_value(rdc_handle_t p_rdc_handle, uint32_t gpu_index, uint32_t field, rdc_field_value *value)	27
4.1.5.26	rdc_field_get_value_since(rdc_handle_t p_rdc_handle, uint32_t gpu_index, uint32_t field, uint64_t since_time_stamp, uint64_t *next_since_time_stamp, rdc_field_value *value)	27
4.1.5.27	rdc_field_unwatch(rdc_handle_t p_rdc_handle, rdc_gpu_group_t group_id, rdc← _field_grp_t field_group_id)	27
4.1.5.28	rdc_status_string(rdc_status_t status)	28
4.1.5.29	field_id_string(uint32_t field_id)	28

29

Index

# **Chapter 1**

# **Data Structure Index**

## 1.1 Data Structures

Here are the data structures with brief descriptions:

rdc_device_attributes_t	
Represents attributes corresponding to a device	5
rdc_field_group_info_t	
The structure to store the field group info	5
rdc_field_value	
The structure to store the field value	6
rdc_gpu_usage_info_t	
The structure to hold the GPU usage information	7
rdc_group_info_t	
The structure to store the group info	8
rdc_job_group_info_t	
The structure to store the job info	8
rdc_job_info_t	
The structure to hold the job stats	9
rdc_stats_summary_t	
The structure to store summary of data	0

2 Data Structure Index

## **Chapter 2**

## File Index

## 2.1 File List

Here is a list of all documented files with brief descriptions:

rdc.h

The rocm\_rdc library api is new, and therefore subject to change either at the ABI or API level. Instead of marking every function prototype as "unstable", we are instead saying the API is unstable (i.e., changes are possible) while the major version remains 0. This means that if the API/ABI changes, we will not increment the major version to 1. Once the ABI stabilizes, we will increment the major version to 1, and thereafter increment it on all ABI breaks . . . . . . . . . .

11

File Index

## **Chapter 3**

## **Data Structure Documentation**

## 3.1 rdc\_device\_attributes\_t Struct Reference

Represents attributes corresponding to a device.

```
#include <rdc.h>
```

#### **Data Fields**

• char device\_name [RDC\_MAX\_STR\_LENGTH]

Name of the device.

## 3.1.1 Detailed Description

Represents attributes corresponding to a device.

The documentation for this struct was generated from the following file:

rdc.h

## 3.2 rdc\_field\_group\_info\_t Struct Reference

The structure to store the field group info.

```
#include <rdc.h>
```

#### **Data Fields**

- uint32\_t count
  - count of fields in the group
- char group\_name [RDC\_MAX\_STR\_LENGTH]

field group name

• uint32\_t field\_ids [RDC\_MAX\_FIELD\_IDS\_PER\_FIELD\_GROUP]

#### 3.2.1 Detailed Description

The structure to store the field group info.

#### 3.2.2 Field Documentation

```
3.2.2.1 uint32_t rdc_field_group_info_t::field_ids[RDC_MAX_FIELD_IDS_PER_FIELD_GROUP]
```

The list of fields in the group

The documentation for this struct was generated from the following file:

· rdc.h

## 3.3 rdc\_field\_value Struct Reference

The structure to store the field value.

```
#include <rdc.h>
```

#### **Data Fields**

```
    uint32_t field_id
        The field id of the value.
    int status
        RDC_ST_OK or error status.
    uint64_t ts
        Timestamp in usec since 1970.
    rdc_field_type_t type
        The field type.
    union {
        int64_t I_int
        double dbl
        char str [RDC_MAX_STR_LENGTH]
    } value
```

### 3.3.1 Detailed Description

The structure to store the field value.

#### 3.3.2 Field Documentation

```
3.3.2.1 union { ... } rdc_field_value::value
```

Value of the field. Value type depends on the field type.

The documentation for this struct was generated from the following file:

• rdc.h

## 3.4 rdc\_gpu\_usage\_info\_t Struct Reference

The structure to hold the GPU usage information.

```
#include <rdc.h>
```

#### **Data Fields**

```
• uint32_t gpu_id
```

GPU\_ID\_INVALID for summary information.

uint64\_t start\_time

The time to start the watching.

• uint64\_t end\_time

The time to stop the watching.

• uint64\_t energy\_consumed

GPU Energy consumed.

• uint64\_t ecc\_correct

Correctable errors.

uint64\_t ecc\_uncorrect

Uncorrtable errors.

rdc\_stats\_summary\_t pcie\_tx

Bytes sent over PCIe stats.

• rdc\_stats\_summary\_t pcie\_rx

Bytes received over PCIe stats.

rdc\_stats\_summary\_t power\_usage

GPU Power usage stats.

rdc\_stats\_summary\_t gpu\_clock

GPU Clock speed stats.

• rdc\_stats\_summary\_t memory\_clock

Mem. Clock speed stats.

rdc\_stats\_summary\_t gpu\_utilization

GPU Utilization stats.

• rdc\_stats\_summary\_t gpu\_temperature

GPU temperature stats.

• uint64\_t max\_gpu\_memory\_used

Maximum GPU memory used.

rdc\_stats\_summary\_t memory\_utilization

Memory Utilization statistics.

#### 3.4.1 Detailed Description

The structure to hold the GPU usage information.

The documentation for this struct was generated from the following file:

· rdc.h

## 3.5 rdc\_group\_info\_t Struct Reference

The structure to store the group info.

```
#include <rdc.h>
```

#### **Data Fields**

- unsigned int count
  - count of GPUs in the group
- char group\_name [RDC\_MAX\_STR\_LENGTH] group name
- uint32\_t entity\_ids [RDC\_GROUP\_MAX\_ENTITIES]

### 3.5.1 Detailed Description

The structure to store the group info.

#### 3.5.2 Field Documentation

```
3.5.2.1 uint32_t rdc_group_info_t::entity_ids[RDC_GROUP_MAX_ENTITIES]
```

The list of entities in the group

The documentation for this struct was generated from the following file:

• rdc.h

## 3.6 rdc\_job\_group\_info\_t Struct Reference

The structure to store the job info.

```
#include <rdc.h>
```

#### **Data Fields**

```
    char job_id [RDC_MAX_STR_LENGTH]
        job id
    rdc_gpu_group_t group_id
        group name
    uint64_t start_time
        job start time
    uint64_t stop_time
        job stop time
```

### 3.6.1 Detailed Description

The structure to store the job info.

The documentation for this struct was generated from the following file:

· rdc.h

## 3.7 rdc\_job\_info\_t Struct Reference

The structure to hold the job stats.

```
#include <rdc.h>
```

#### **Data Fields**

```
    uint32_t num_gpus
        Number of GPUs used by job.

    rdc_gpu_usage_info_t summary
```

rdc\_gpu\_usage\_info\_t gpus [16]

Job usage summary staticstics by GPU.

#### 3.7.1 Detailed Description

The structure to hold the job stats.

#### 3.7.2 Field Documentation

```
3.7.2.1 rdc_gpu_usage_info_t rdc_job_info_t::summary
```

Job usage summary statistics (overall)

The documentation for this struct was generated from the following file:

• rdc.h

## 3.8 rdc\_stats\_summary\_t Struct Reference

The structure to store summary of data.

```
#include <rdc.h>
```

#### **Data Fields**

• uint64\_t max\_value

Maximum value measured.

uint64\_t min\_value

Minimum value measured.

uint64\_t average

Average value measured.

### 3.8.1 Detailed Description

The structure to store summary of data.

The documentation for this struct was generated from the following file:

· rdc.h

## **Chapter 4**

## **File Documentation**

#### 4.1 rdc.h File Reference

The rocm\_rdc library api is new, and therefore subject to change either at the ABI or API level. Instead of marking every function prototype as "unstable", we are instead saying the API is unstable (i.e., changes are possible) while the major version remains 0. This means that if the API/ABI changes, we will not increment the major version to 1. Once the ABI stabilizes, we will increment the major version to 1, and thereafter increment it on all ABI breaks.

```
#include <cstdint>
```

#### **Data Structures**

• struct rdc\_device\_attributes\_t

Represents attributes corresponding to a device.

• struct rdc\_group\_info\_t

The structure to store the group info.

• struct rdc\_stats\_summary\_t

The structure to store summary of data.

· struct rdc\_gpu\_usage\_info\_t

The structure to hold the GPU usage information.

struct rdc\_job\_info\_t

The structure to hold the job stats.

• struct rdc\_field\_value

The structure to store the field value.

struct rdc\_field\_group\_info\_t

The structure to store the field group info.

· struct rdc\_job\_group\_info\_t

The structure to store the job info.

#### **Macros**

• #define GPU\_ID\_INVALID -1

ID used to represent an invalid GPU.

• #define RDC\_GROUP\_ALL\_GPUS -1000

Used to specify all GPUs.

• #define RDC\_JOB\_STATS\_FIELDS -1000

Used to specify all stats fields.

• #define RDC\_MAX\_STR\_LENGTH 256

The max rdc field string length.

• #define RDC\_GROUP\_MAX\_ENTITIES 64

The max entities in a group.

• #define RDC MAX NUM DEVICES 16

Max number of GPUs supported by RDC.

• #define RDC\_MAX\_FIELD\_IDS\_PER\_FIELD\_GROUP 128

The max fields in a field group.

• #define RDC\_MAX\_NUM\_GROUPS 64

The max number of groups.

• #define RDC\_MAX\_NUM\_FIELD\_GROUPS 64

The max number of the field groups.

- #define RDC FI GPU MEMORY USAGE 525
- #define RDC\_FI\_GPU\_MEMORY\_TOTAL 580
- #define RDC\_FI\_POWER\_USAGE 155
- #define RDC\_FI\_GPU\_CLOCK 100
- #define RDC\_FI\_MEM\_CLOCK 101
- #define RDC\_FI\_PCIE\_TX 200
- #define RDC\_FI\_PCIE\_RX 201
- #define RDC\_FI\_GPU\_UTIL 203
- #define RDC FI ECC CORRECT TOTAL 312
- #define RDC\_FI\_ECC\_UNCORRECT\_TOTAL 313
- #define RDC\_FI\_MEMORY\_TEMP 140
- #define RDC FI GPU TEMP 150
- #define RDC\_FI\_GPU\_COUNT 4
- #define RDC\_FI\_DEV\_NAME 50

## **Typedefs**

typedef void \* rdc handle t

handlers used in various rdc calls

• typedef uint32\_t rdc\_gpu\_group\_t

GPU Group ID type.

• typedef uint32\_t rdc\_field\_grp\_t

Field group ID type.

#### **Enumerations**

enum rdc\_status\_t {
 RDC\_ST\_OK = 0, RDC\_ST\_NOT\_SUPPORTED, RDC\_ST\_MSI\_ERROR, RDC\_ST\_FAIL\_LOAD\_MODU
 LE,
 RDC\_ST\_INVALID\_HANDLER, RDC\_ST\_BAD\_PARAMETER, RDC\_ST\_NOT\_FOUND, RDC\_ST\_CON
 FLICT,
 RDC\_ST\_CLIENT\_ERROR, RDC\_ST\_ALREADY\_EXIST, RDC\_ST\_MAX\_LIMIT }

Error codes returned by rocm\_rdc\_lib functions.

enum rdc\_operation\_mode\_t { RDC\_OPERATION\_MODE\_AUTO = 0, RDC\_OPERATION\_MODE\_MAN ← UAL }

rdc operation mode rdc can run in auto mode where background threads will collect metrics. When run in manual mode, the user needs to periodically call rdc\_field\_update\_all for data collection.

enum rdc\_group\_type\_t { RDC\_GROUP\_DEFAULT = 0, RDC\_GROUP\_EMPTY }

type of GPU group

• enum rdc\_field\_type\_t { INTEGER = 0, DOUBLE, STRING, BLOB }

the type stored in the filed value

#### **Functions**

• rdc status t rdc init (uint64 t init flags)

Initialize ROCm RDC.

• rdc\_status\_t rdc\_shutdown ()

Shutdown ROCm RDC.

rdc\_status\_t rdc\_start\_embedded (rdc\_operation\_mode\_t op\_mode, rdc\_handle\_t \*p\_rdc\_handle)

Start embedded RDC agent within this process.

rdc\_status\_t rdc\_stop\_embedded (rdc\_handle\_t p\_rdc\_handle)

Stop embedded RDC agent.

• rdc\_status\_t rdc\_connect (const char \*ipAndPort, rdc\_handle\_t \*p\_rdc\_handle, const char \*root\_ca, const char \*client\_cert, const char \*client\_key)

Connect to rdcd daemon.

• rdc status t rdc disconnect (rdc handle t p rdc handle)

Disconnect from rdcd daemon.

rdc\_status\_t rdc\_job\_start\_stats (rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, const char job\_
 id[64], uint64\_t update\_freq)

Request the RDC to watch the job stats.

rdc\_status\_t rdc\_job\_get\_stats (rdc\_handle\_t p\_rdc\_handle, const char job\_id[64], rdc\_job\_info\_t \*p\_job\_
info)

Get the stats of the job using the job id.

rdc\_status\_t rdc\_job\_stop\_stats (rdc\_handle\_t p\_rdc\_handle, const char job\_id[64])

Request RDC to stop watching the stats of the job.

rdc\_status\_t rdc\_job\_remove (rdc\_handle\_t p\_rdc\_handle, const char job\_id[64])

Request RDC to stop tracking the job given by job\_id.

rdc\_status\_t rdc\_job\_remove\_all (rdc\_handle\_t p\_rdc\_handle)

Request RDC to stop tracking all the jobs.

• rdc\_status\_t rdc\_field\_update\_all (rdc\_handle\_t p\_rdc\_handle, uint32\_t wait\_for\_update)

Request RDC to update all fields to be watched.

 rdc\_status\_t rdc\_device\_get\_all (rdc\_handle\_t p\_rdc\_handle, uint32\_t gpu\_index\_list[RDC\_MAX\_NUM\_D← EVICES], uint32\_t \*count)

Get indexes corresponding to all the devices on the system.

rdc\_status\_t rdc\_device\_get\_attributes (rdc\_handle\_t p\_rdc\_handle, uint32\_t gpu\_index, rdc\_device\_
 attributes\_t \*p\_rdc\_attr)

Gets device attributes corresponding to the gpu\_index.

 rdc\_status\_t rdc\_group\_gpu\_create (rdc\_handle\_t p\_rdc\_handle, rdc\_group\_type\_t type, const char \*group\_name, rdc\_gpu\_group\_t \*p\_rdc\_group\_id)

Create a group contains multiple GPUs.

rdc\_status\_t rdc\_group\_gpu\_add (rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, uint32\_t gpu\_
 index)

Add a GPU to the group.

rdc\_status\_t rdc\_group\_gpu\_get\_info (rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t p\_rdc\_group\_id, rdc
group info t \*p rdc group info)

Get information about a GPU group.

rdc\_status\_t rdc\_group\_get\_all\_ids (rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id\_list[], uint32\_t \*count)

Used to get information about all GPU groups in the system.

- rdc\_status\_t rdc\_group\_gpu\_destroy (rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t p\_rdc\_group\_id)

  Destroy GPU group represented by p\_rdc\_group\_id.
- rdc\_status\_t rdc\_group\_field\_create (rdc\_handle\_t p\_rdc\_handle, uint32\_t num\_field\_ids, uint32\_t \*field\_ids, const char \*field\_group\_name, rdc\_field\_grp\_t \*rdc\_field\_group\_id)

create a group of fields

rdc\_status\_t rdc\_group\_field\_get\_info (rdc\_handle\_t p\_rdc\_handle, rdc\_field\_grp\_t rdc\_field\_group\_id, rdc
 field\_group\_info\_t \*field\_group\_info)

Get information about a field group.

 rdc\_status\_t rdc\_group\_field\_get\_all\_ids (rdc\_handle\_t p\_rdc\_handle, rdc\_field\_grp\_t field\_group\_id\_list[], uint32 t \*count)

Used to get information about all field groups in the system.

- rdc\_status\_t rdc\_group\_field\_destroy (rdc\_handle\_t p\_rdc\_handle, rdc\_field\_grp\_t rdc\_field\_group\_id)

  Destroy field group represented by rdc\_field\_group\_id.
- rdc\_status\_t rdc\_field\_watch (rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, rdc\_field\_grp\_t field 
  \_group\_id, uint64\_t update\_freq, double max\_keep\_age, uint32\_t max\_keep\_samples)

Request the RDC start recording updates for a given field collection.

rdc\_status\_t rdc\_field\_get\_latest\_value (rdc\_handle\_t p\_rdc\_handle, uint32\_t gpu\_index, uint32\_t field, rdc
field\_value \*value)

Request a latest cached field of a GPU.

• rdc\_status\_t rdc\_field\_get\_value\_since (rdc\_handle\_t p\_rdc\_handle, uint32\_t gpu\_index, uint32\_t field, uint64\_t since\_time\_stamp, uint64\_t \*next\_since\_time\_stamp, rdc\_field\_value \*value)

Request a history cached field of a GPU.

rdc\_status\_t rdc\_field\_unwatch (rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, rdc\_field\_grp\_
 t field\_group\_id)

Stop record updates for a given field collection.

const char \* rdc\_status\_string (rdc\_status\_t status)

Get a description of a provided RDC error status.

const char \* field\_id\_string (uint32\_t field\_id)

Get the name of a field.

#### 4.1.1 Detailed Description

The rocm\_rdc library api is new, and therefore subject to change either at the ABI or API level. Instead of marking every function prototype as "unstable", we are instead saying the API is unstable (i.e., changes are possible) while the major version remains 0. This means that if the API/ABI changes, we will not increment the major version to 1. Once the ABI stabilizes, we will increment the major version to 1, and thereafter increment it on all ABI breaks.

Main header file for the ROCm RDC library. All required function, structure, enum, etc. definitions should be defined in this file.

4.1.2 Macro Definition Documentation

4.1.2.1 #define RDC\_FI\_GPU\_MEMORY\_USAGE 525

Memory usage of the GPU instance

4.1.2.2 #define RDC\_FI\_GPU\_MEMORY\_TOTAL 580

Total memory of the GPU instance

4.1.2.3 #define RDC\_FI\_POWER\_USAGE 155

Power usage for the device

4.1.2.4 #define RDC\_FI\_GPU\_CLOCK 100

The current clock for the GPU

4.1.2.5 #define RDC\_FI\_MEM\_CLOCK 101

Clock for the memory

4.1.2.6 #define RDC\_FI\_PCIE\_TX 200

PCIe Tx utilization information

4.1.2.7 #define RDC\_FI\_PCIE\_RX 201

PCIe Rx utilization information

4.1.2.8 #define RDC\_FI\_GPU\_UTIL 203

**GPU Utilization** 

4.1.2.9 #define RDC\_FI\_ECC\_CORRECT\_TOTAL 312

Accumulated correctable ECC errors

4.1.2.10 #define RDC\_FI\_ECC\_UNCORRECT\_TOTAL 313

Accumulated uncorrectable ECC errors

4.1.2.11 #define RDC\_FI\_MEMORY\_TEMP 140

Memory temperature for the device

4.1.2.12 #define RDC\_FI\_GPU\_TEMP 150

Current temperature for the device

4.1.2.13 #define RDC\_FI\_GPU\_COUNT 4

GPU count in the system

4.1.2.14 #define RDC\_FI\_DEV\_NAME 50

Name of the device

#### 4.1.3 Typedef Documentation

4.1.3.1 typedef void\* rdc\_handle\_t

handlers used in various rdc calls

Handle used for an RDC session

#### 4.1.4 Enumeration Type Documentation

4.1.4.1 enum rdc\_status\_t

Error codes returned by rocm\_rdc\_lib functions.

#### Enumerator

RDC\_ST\_OK Success.

RDC\_ST\_NOT\_SUPPORTED Not supported feature.

RDC\_ST\_MSI\_ERROR The MSI library error.

RDC\_ST\_FAIL\_LOAD\_MODULE Fail to load the library.

RDC\_ST\_INVALID\_HANDLER Invalid handler.

RDC\_ST\_BAD\_PARAMETER A parameter is invalid.

RDC\_ST\_NOT\_FOUND Cannot find the value.

RDC\_ST\_CONFLICT Conflict with current state.

RDC\_ST\_CLIENT\_ERROR The RDC client error.

RDC\_ST\_ALREADY\_EXIST The item already exists.

RDC\_ST\_MAX\_LIMIT Max limit recording for the object.

4.1.4.2 enum rdc\_group\_type\_t

type of GPU group

#### Enumerator

**RDC\_GROUP\_DEFAULT** All GPUs on the Node. **RDC\_GROUP\_EMPTY** Empty group.

#### 4.1.5 Function Documentation

4.1.5.1 rdc\_status\_t rdc\_init ( uint64\_t init\_flags )

Initialize ROCm RDC.

When called, this initializes internal data structures, including those corresponding to sources of information that RDC provides. This must be called before rdc\_start\_embedded() or rdc\_connect()

#### **Parameters**

	in	init_flags	init_flags Bit flags that tell RDC how to initialize.
--	----	------------	---

#### Return values

RDC_ST_OK is returned upon successful ca
--

4.1.5.2 rdc\_status\_t rdc\_shutdown()

Shutdown ROCm RDC.

Do any necessary clean up.

4.1.5.3 rdc\_status\_t rdc\_start\_embedded ( rdc\_operation\_mode\_t op\_mode, rdc\_handle\_t \* p\_rdc\_handle )

Start embedded RDC agent within this process.

The RDC is loaded as library so that it does not require rdcd daemon. In this mode, the user has to periodically call rdc\_field\_update\_all() when op\_mode is RDC\_OPERATION\_MODE\_MANUAL, which tells RDC to collect the stats.

### Parameters

in	op_mode	Operation modes. When RDC_OPERATION_MODE_AUTO, RDC schedules background task to collect the stats. When RDC_OPERATION_MODE_MANUAL, the user needs to call rdc_field_update_all() periodically.
in,out	p_rdc_handle	Caller provided pointer to rdc_handle_t. Upon successful call, the value will contain the handler for following API calls.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.4 rdc\_status\_t rdc\_stop\_embedded ( rdc\_handle\_t p\_rdc\_handle )

Stop embedded RDC agent.

Stop the embedded RDC agent, and p\_rdc\_handle becomes invalid after this call.

#### **Parameters**

i	.n	p_rdc_handle	The RDC handler that come from rdc_start_embedded().	]
---	----	--------------	--	---

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.5 rdc\_status\_t rdc\_connect ( const char \* ipAndPort, rdc\_handle\_t \* p\_rdc\_handle, const char \* root\_ca, const char \* client\_cert, const char \* client\_key )

Connect to rdcd daemon.

This method is used to connect to a remote stand-alone rdcd daemon.

#### **Parameters**

in	ipAndPort	The IP and port of the remote rdcd. The ipAndPort can be specified in this x.x.x.x:yyyy format, where x.x.x.x is the IP address and yyyy is the port.
in,out	p_rdc_handle	Caller provided pointer to rdc_handle_t. Upon successful call, the value will contain the handler for following API calls.
in	root_ca	The root CA stored in the string in pem format. Set it as nullptr if the communication is not encrypted.
in	client_cert	The client certificate stored in the string in pem format. Set it as nullptr if the communication is not encrypted.
in	client_key	The client key stored in the string in pem format. Set it as nullptr if the communication is not encrypted.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.6 rdc\_status\_t rdc\_disconnect ( rdc\_handle\_t p\_rdc\_handle )

Disconnect from rdcd daemon.

4.1 rdc.h File Reference 19 Disconnect from rdcd daemon, and p\_rdc\_handle becomes invalid after this call.

#### **Parameters**

in	p_rdc_handle	The RDC handler that come from rdc_connect().
----	--------------	---

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.7 rdc\_status\_t rdc\_job\_start\_stats ( rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, const char job\_id[64], uint64\_t update\_freq )

Request the RDC to watch the job stats.

This should be executed as part of job prologue. The summary job stats can be retrieved using rdc\_job\_get\_color stats(). In RDC\_OPERATION\_MODE\_MANUAL, user must call rdc\_field\_update\_all(1) at least once, before call rdc\_job\_get\_stats()

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	group_id	The group of GPUs to be watched.
in	job_id	The name of the job.
in	update_freq	How often to update this field in usec.

#### **Return values**

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.8 rdc\_status\_t rdc\_job\_get\_stats ( rdc\_handle\_t p\_rdc\_handle, const char job\_id[64], rdc\_job\_info\_t \* p\_job\_info )

Get the stats of the job using the job id.

The stats can be retrieved at any point when the job is in process.

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	job_id	The name of the job.
in,out	p_job_info	Caller provided pointer to rdc_job_info_t. Upon successful call, the value will contain the stats of the job.

#### **Return values**

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.9 rdc\_status\_t rdc\_job\_stop\_stats ( rdc\_handle\_t p\_rdc\_handle, const char job\_id[64] )

Request RDC to stop watching the stats of the job.

This should be execute as part of job epilogue. The job ld remains available to view the stats at any point. You must call rdc\_watch\_job\_fields() before this call.

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	job_id	The name of the job.

#### **Return values**

4.1.5.10 rdc\_status\_t rdc\_job\_remove ( rdc\_handle\_t p\_rdc\_handle, const char job\_id[64] )

Request RDC to stop tracking the job given by job\_id.

After this call, you will no longer be able to call rdc\_job\_get\_stats() on this job\_id. But you will be able to reuse the job\_id after this call.

#### **Parameters**

	in	p_rdc_handle	The RDC handler.
ĺ	in	job_id	The name of the job.

#### Return values

RDC_ST_OK	is returned upon successful call.

4.1.5.11 rdc\_status\_t rdc\_job\_remove\_all ( rdc\_handle\_t p\_rdc\_handle )

Request RDC to stop tracking all the jobs.

After this call, you will no longer be able to call <a href="rdc\_job\_get\_stats">rdc\_job\_get\_stats</a>() on any job id. But you will be able to reuse the any previous used job id after this call.

#### **Parameters**

in	p_rdc_handle	The RDC handler.
----	--------------	------------------

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.12 rdc\_status\_t rdc\_field\_update\_all ( rdc\_handle\_t p\_rdc\_handle, uint32\_t wait\_for\_update )

Request RDC to update all fields to be watched.

In RDC\_OPERATION\_MODE\_MANUAL, the user must call this method periodically.

#### **Parameters**

-	in	p_rdc_handle	The RDC handler.
	in	wait_for_update	Whether or not to wait for the update loop to complete before returning to the caller
			1=wait. 0=do not wait.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.13 rdc\_status\_t rdc\_device\_get\_all ( rdc\_handle\_t p\_rdc\_handle, uint32\_t gpu\_index\_list[RDC\_MAX\_NUM\_DEVIC← ES], uint32\_t \* count )

Get indexes corresponding to all the devices on the system.

Indexes represents RDC GPU Id corresponding to each GPU on the system and is immutable during the lifespan of the engine. The list should be queried again if the engine is restarted.

#### **Parameters**

in	p_rdc_handle	The RDC handler.
out	gpu_index_list	Array reference to fill GPU indexes present on the system.
out	count	Number of GPUs returned in gpu_index_list.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.14 rdc\_status\_t rdc\_device\_get\_attributes ( rdc\_handle\_t  $p_rdc_handle$ , uint32\_t  $gpu_index$ , rdc\_device\_attributes\_t \*  $p_rdc_attr$ )

Gets device attributes corresponding to the  $gpu\_index$ .

Fetch the attributes, such as device name, of a GPU.

#### Parameters

in	p_rdc_handle	The RDC handler.
in	gpu_index	GPU index corresponding to which the attributes should be fetched
out	p_rdc_attr	GPU attribute corresponding to the gpu_index.

#### Return values

RDC_ST_OK is returned upon successful call.
---

4.1.5.15 rdc\_status\_t rdc\_group\_gpu\_create ( rdc\_handle\_t p\_rdc\_handle, rdc\_group\_type\_t type, const char \* group\_name, rdc\_gpu\_group\_t \* p\_rdc\_group\_id )

Create a group contains multiple GPUs.

This method can create a group contains multiple GPUs. Instead of executing an operation separately for each GPU, the RDC group enables the user to execute same operation on all the GPUs present in the group as a single API call.

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	type	The type of the group. RDC_GROUP_DEFAULT includes all the GPUs on the node, and RDC_GROUP_EMPTY creates an empty group.
in	group_name	The group name specified as NULL terminated C String
in, out	p_rdc_group↔ _id	Caller provided pointer to rdc_gpu_group_t. Upon successful call, the value will contain the group id for following group API calls.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.16 rdc\_status\_t rdc\_group\_gpu\_add ( rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, uint32\_t gpu\_index )

Add a GPU to the group.

This method can add a GPU to the group

#### Parameters

in	p_rdc_handle	The RDC handler.
in	group_id	The group id to which the GPU will be added.
in	gpu_index	The GPU index to be added to the group.

#### Return values

RDC_ST_OK	is returned upon successful call.

4.1.5.17 rdc\_status\_t rdc\_group\_gpu\_get\_info ( rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t p\_rdc\_group\_id, rdc\_group\_info\_t \* p\_rdc\_group\_info )

Get information about a GPU group.

Get detail information about a GPU group created by rdc\_group\_gpu\_create

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	p_rdc_group_id	The GPU group handler created by rdc_group_gpu_create
out	p_rdc_group_info	The information of the GPU group p_rdc_group_id.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.18 rdc\_status\_t rdc\_group\_get\_all\_ids ( rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id\_list[], uint32\_t \* count )

Used to get information about all GPU groups in the system.

Get the list of GPU group ids in the system.

#### Parameters

in	p_rdc_handle	The RDC handler.
out	group_id_list	Array reference to fill GPU group ids in the system.
out	count	Number of GPU group returned in group_id_list.

#### **Return values**

RDC_ST_OK	is returned upon successful call.

4.1.5.19 rdc\_status\_t rdc\_group\_gpu\_destroy ( rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t p\_rdc\_group\_id )

Destroy GPU group represented by p\_rdc\_group\_id.

Delete the logic group represented by p\_rdc\_group\_id

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	p_rdc_group←	The group id
	_id	

#### Return values

RDC ST OK	is returned upon successful call.

4.1.5.20 rdc\_status\_t rdc\_group\_field\_create ( rdc\_handle\_t p\_rdc\_handle, uint32\_t num\_field\_ids, uint32\_t \* field\_ids, const char \* field\_group\_name, rdc\_field\_group\_t \* rdc\_field\_group\_id )

create a group of fields

The user can create a group of fields and perform an operation on a group of fields at once.

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	num_field_ids	Number of field IDs that are being provided in field_ids.
in	field_ids	Field IDs to be added to the newly-created field group.
in	field_group_name	Unique name for this group of fields.
out	rdc_field_group⇔	Handle to the newly-created field group
	_id	

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.21 rdc\_status\_t rdc\_group\_field\_get\_info ( rdc\_handle\_t p\_rdc\_handle, rdc\_field\_group\_i rdc\_field\_group\_info\_t \* field\_group\_info\_)

Get information about a field group.

Get detail information about a field group created by rdc\_group\_field\_create

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	rdc_field_group↔	The field group handler created by rdc_group_field_create
	_id	
out	field_group_info	The information of the field group rdc_field_group_id.

#### Return values

RDC_ST_OK	is returned upon successful call.

4.1.5.22 rdc\_status\_t rdc\_group\_field\_get\_all\_ids ( rdc\_handle\_t p\_rdc\_handle, rdc\_field\_grp\_t field\_group\_id\_list[], uint32\_t \* count )

Used to get information about all field groups in the system.

Get the list of field group ids in the system.

#### **Parameters**

	in	p_rdc_handle	The RDC handler.
	out	field_group_id_list	Array reference to fill field group ids in the system.
ĺ	out	count	Number of field group returned in field_group_id_list.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.23 rdc\_status\_t rdc\_group\_field\_destroy ( rdc\_handle\_t p\_rdc\_handle, rdc\_field\_grp\_t rdc\_field\_group\_id )

Destroy field group represented by rdc\_field\_group\_id.

Delete the logic group represented by rdc\_field\_group\_id

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	rdc_field_group⇔	The field group id
	_id	

#### **Return values**

4.1.5.24 rdc\_status\_t rdc\_field\_watch ( rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, rdc\_field\_grp\_t field\_group\_id, uint64\_t update\_freq, double max\_keep\_age, uint32\_t max\_keep\_samples )

Request the RDC start recording updates for a given field collection.

Note that the first update of the field will not occur until the next field update cycle. To force a field update cycle, user must call rdc\_field\_update\_all(1)

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in group_id The group of GPUs to be watched.		The group of GPUs to be watched.
in	field_group_id	The collection of fields to record
in	in update_freq How often to update fields in usec.	
in	max_keep_age	How long to keep data for fields in seconds.
in	max_keep_samples	Maximum number of samples to keep. 0=no limit.

#### Return values

RDC ST OK	is returned upon successful call.

4.1.5.25 rdc\_status\_t rdc\_field\_get\_latest\_value ( rdc\_handle\_t p\_rdc\_handle, uint32\_t gpu\_index, uint32\_t field, rdc\_field\_value \* value )

Request a latest cached field of a GPU.

Note that the field can be cached after called rdc\_field\_watch

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	gpu_index	The GPU index.
in	field	The field id
out	value	The field value got from cache.

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.26 rdc\_status\_t rdc\_field\_get\_value\_since ( rdc\_handle\_t p\_rdc\_handle, uint32\_t gpu\_index, uint32\_t field, uint64\_t since\_time\_stamp, uint64\_t \* next\_since\_time\_stamp, rdc\_field\_value \* value )

Request a history cached field of a GPU.

Note that the field can be cached after called rdc\_field\_watch

### Parameters

in	p_rdc_handle	The RDC handler.
in	gpu_index	The GPU index.
in	field	The field id
in	since_time_stamp	Timestamp to request values since in usec since 1970.
out	next_since_time_stamp	Timestamp to use for sinceTimestamp on next call to this function
out	value	The field value got from cache.

#### Return values

RDC_ST_OK is returned upon successful
---------------------------------------

4.1.5.27 rdc\_status\_t rdc\_field\_unwatch ( rdc\_handle\_t p\_rdc\_handle, rdc\_gpu\_group\_t group\_id, rdc\_field\_grop\_t field\_group\_id )

Stop record updates for a given field collection.

The cache of those fields will not be updated after this call

#### **Parameters**

in	p_rdc_handle	The RDC handler.
in	group_id	The GPU group id.
in	field_group⊷	The field group id.
	_id	

#### Return values

RDC_ST_OK	is returned upon successful call.
-----------	-----------------------------------

4.1.5.28 const char\* rdc\_status\_string ( rdc\_status\_t status )

Get a description of a provided RDC error status.

return the string in human readable format.

#### **Parameters**

in status The	RDC status.
---------------	-------------

#### Return values

The	string to describe the RDC status.
-----	------------------------------------

4.1.5.29 const char\* field\_id\_string ( uint32\_t field\_id )

Get the name of a field.

return the string in human readable format.

#### **Parameters**

	in	field⇔	The field id.
l		id	

#### Return values

The	string to describe the field.

# Index

entity_ids	RDC_ST_INVALID_HANDLER
rdc_group_info_t, 8	rdc.h, 16
	RDC_ST_MAX_LIMIT
field_id_string	rdc.h, 16
rdc.h, 28	RDC_ST_MSI_ERROR
field_ids	rdc.h, 16
rdc_field_group_info_t, 6	RDC_ST_NOT_FOUND
DDG EL DEV MANE	rdc.h, 16
RDC_FI_DEV_NAME	RDC_ST_NOT_SUPPORTED
rdc.h, 16	rdc.h, 16
RDC_FI_ECC_CORRECT_TOTAL	RDC_ST_OK
rdc.h, 15	rdc.h, 16
RDC_FI_ECC_UNCORRECT_TOTAL	rdc.h, 11
rdc.h, 15	field_id_string, 28
RDC_FI_GPU_CLOCK	RDC_FI_DEV_NAME, 16
rdc.h, 15	RDC_FI_ECC_CORRECT_TOTAL, 15
RDC_FI_GPU_COUNT	RDC_FI_ECC_UNCORRECT_TOTAL, 15
rdc.h, 16	RDC FI GPU CLOCK, 15
RDC_FI_GPU_MEMORY_TOTAL	RDC_FI_GPU_COUNT, 16
rdc.h, 15	RDC_FI_GPU_MEMORY_TOTAL, 15
RDC_FI_GPU_MEMORY_USAGE	RDC_FI_GPU_MEMORY_USAGE, 15
rdc.h, 15	RDC_FI_GPU_TEMP, 16
RDC_FI_GPU_TEMP	RDC_FI_GPU_UTIL, 15
rdc.h, 16	RDC FI MEM CLOCK, 15
RDC_FI_GPU_UTIL	RDC_FI_MEMORY_TEMP, 15
rdc.h, 15	
RDC_FI_MEM_CLOCK	RDC_FI_PCIE_RX, 15
rdc.h, 15	RDC_FI_PCIE_TX, 15
RDC_FI_MEMORY_TEMP	RDC_FI_POWER_USAGE, 15
rdc.h, 15	RDC_GROUP_DEFAULT, 17
RDC_FI_PCIE_RX	RDC_GROUP_EMPTY, 17
rdc.h, 15	RDC_ST_ALREADY_EXIST, 16
RDC_FI_PCIE_TX	RDC_ST_BAD_PARAMETER, 16
rdc.h, 15	RDC_ST_CLIENT_ERROR, 16
RDC_FI_POWER_USAGE	RDC_ST_CONFLICT, 16
rdc.h, 15	RDC_ST_FAIL_LOAD_MODULE, 16
RDC_GROUP_DEFAULT	RDC_ST_INVALID_HANDLER, 16
rdc.h, 17	RDC_ST_MAX_LIMIT, 16
RDC_GROUP_EMPTY	RDC_ST_MSI_ERROR, 16
rdc.h, 17	RDC_ST_NOT_FOUND, 16
RDC_ST_ALREADY_EXIST	RDC_ST_NOT_SUPPORTED, 16
rdc.h, 16	RDC_ST_OK, 16
RDC_ST_BAD_PARAMETER	rdc_connect, 18
rdc.h, 16	rdc_device_get_all, 22
RDC_ST_CLIENT_ERROR	rdc_device_get_attributes, 22
rdc.h, 16	rdc disconnect, 18
RDC_ST_CONFLICT	rdc_field_get_latest_value, 27
rdc.h, 16	rdc_field_get_value_since, 27
RDC_ST_FAIL_LOAD_MODULE	rdc_field_unwatch, 27
rdc.h, 16	rdc_field_update_all, 22

30 INDEX

rdc_field_watch, 26	rdc.h, 23
rdc_group_field_create, 25	rdc_group_gpu_create
rdc_group_field_destroy, 26	rdc.h, 23
rdc_group_field_get_all_ids, 25	rdc_group_gpu_destroy
rdc_group_field_get_info, 25	rdc.h, 24
rdc_group_get_all_ids, 24	rdc_group_gpu_get_info
rdc_group_gpu_add, 23	rdc.h, 23
rdc_group_gpu_create, 23	rdc_group_info_t, 8
rdc_group_gpu_destroy, 24	entity_ids, 8
rdc_group_gpu_get_info, 23	rdc_group_type_t
rdc_group_type_t, 16	rdc.h, 16
rdc_group_type_t, 10	rdc handle t
<u> </u>	rdc.h, 16
rdc_init, 17	rdc_init
rdc_job_get_stats, 20	rdc.h, 17
rdc_job_remove, 21	rdc_job_get_stats
rdc_job_remove_all, 21	rdc.h, 20
rdc_job_start_stats, 20	
rdc_job_stop_stats, 20	rdc_job_group_info_t, 8
rdc_shutdown, 17	rdc_job_info_t, 9
rdc_start_embedded, 17	summary, 9
rdc_status_string, 28	rdc_job_remove
rdc_status_t, 16	rdc.h, 21
rdc_stop_embedded, 18	rdc_job_remove_all
rdc_connect	rdc.h, 21
rdc.h, 18	rdc_job_start_stats
rdc_device_attributes_t, 5	rdc.h, 20
rdc_device_get_all	rdc_job_stop_stats
rdc.h, 22	rdc.h, 20
rdc_device_get_attributes	rdc_shutdown
rdc.h, 22	rdc.h, 17
rdc_disconnect	rdc_start_embedded
rdc.h, 18	rdc.h, 17
rdc_field_get_latest_value	rdc_stats_summary_t, 10
rdc.h, 27	rdc_status_string
rdc_field_get_value_since	rdc.h, 28
rdc.h, 27	rdc_status_t
rdc_field_group_info_t, 5	rdc.h, 16
<del>*</del>	rdc_stop_embedded
field_ids, 6	rdc.h, 18
rdc_field_unwatch	
rdc.h, 27	summary
rdc_field_update_all	rdc_job_info_t, 9
rdc.h, 22	
rdc_field_value, 6	value
value, 7	rdc_field_value, 7
rdc_field_watch	
rdc.h, 26	
rdc_gpu_usage_info_t, 7	
rdc_group_field_create	
rdc.h, 25	
rdc_group_field_destroy	
rdc.h, 26	
rdc_group_field_get_all_ids	
rdc.h, 25	
rdc_group_field_get_info	
rdc.h, 25	
rdc_group_get_all_ids	
rdc.h, 24	
rdc group apu add	
IND DIVUD UDU UUU	