Arrayent Reference Manual

C API Reference

Draft

August 21, 2015 17:48
Generated by Doxygen 1.8.1.1

Contents

1 Main Page **1**

1.1 Summary 1

1.2 Changelog 1

1.3 Details 2

2 Data Structure Index **3**

2.1 Data Structures 3

3 File Index **5**

3.1 File List 5

4 Data Structure Documentation **7**

4.1 arrayent_config_t Struct Reference 7

4.1.1 Field Documentation 7

4.1.1.1 device_can_multi_attribute 7

4.1.1.2 aca_thread_priority 7

4.2 arrayent_net_status_t Struct Reference 8

4.3 arrayent_timestamp_t Struct Reference 8

5 File Documentation **11**

5.1 external/arrayent/source/include/aca.h File Reference 11

5.1.1 Detailed Description 11

5.1.2 Function Documentation 11

5.1.2.1 ArrayentSetConfigDefaults 11

5.1.2.2 ArrayentConfigure 11

5.1.2.3 ArrayentInit 12

5.1.2.4 ArrayentHello 12

5.1.2.5 ArrayentNetStatus 12

5.1.2.6 ArrayentSleep 12

5.1.2.7 ArrayentWake 13

5.1.2.8 ArrayentReset 13

5.1.2.9 ArrayentGetTime 13

5.1.2.10 ArrayentSetProperty 14

5.1.2.11 ArrayentRecvProperty 14
5.1.2.12 ArrayentSetMultiAttribute 14
5.1.2.13 ArrayentRecvMultiAttribute 15
5.1.2.14 ArrayentSendData 15
5.1.2.15 ArrayentRecvData 16
5.1.2.16 ArrayentFactoryResetDevice 16
5.1.2.17 ArrayentDebug 16
5.1.3 Enumeration Type Documentation 17
5.1.3.1 arrayent_return_e 17

Chapter 1

Main Page

1.1 Summary

The Arrayent Connect Agent (ACA) provides connectivity for devices to the Arrayent Connect Cloud.

1.2 Changelog

- 1.5.2.0: 2014/09/12 Fixed potential buffer overflow situations if customer application attempts to transmit over long data buffers. Added state logging over IP. Improved connection testing when heartbeat message is lost. Added support for multi-threaded access to API functions.

- 1.5.1.0: 2014/08/04 Fixed ACA single attribute messages fail when using TCP. Fixed ACA ArrayentNetStatus returns timeout when wifi router WAN port unplugged. Fixed ACA does not return error code on retry failure to set property to cloud. Fixed ACA stuck when cloud resets. Fixed ACA encryption login sequence fails if cloud sends attribute to device during the login process. Added unsigned integer data types support in ACA.

- 1.5.0.0: 2014/06/26 Added to configure message max payload length for POSIX platforms. Enhance ArrayentNetStatus to disclose progress during login sequence. Added AMP ID and SessionEnd ID in arrayent_net_status_t. Fix ACA returns same message twice if two messages arrive before host app calls ArrayentRecvXxx(). Improve ACA error reporting.

- 1.4.3.0: 2014/05/29 Fix ACA intermittently adds garbage characters to end of device name or password.

- 1.4.2.0: 2014/05/23 ACA stops sending heartbeats and goes offline two minutes after restart.

- 1.4.1.0: 2014/04/16 Fix ArrayentInit() hangs when ArrayentConfigure() has not been successfully called. Fix ArrayentNetStatus() is not verified prior to ArrayentSetProperty() then ArrayentSetProperty() returns ARRAYENT_SUCCESS even though the property was not updated in Utility.

- 1.4.0.0: 2014/03/26 Add aca_stack_size field to arrayent_config_t to allow application to configure the stack size for ACA Gateway thread. Add support of data message(200) sub-type. Add ArrayentFactoryResetDevice() to reset device to factory settings. Removed use of message queues for messaging between ACA daemon and ACA APIs. Messages are defined as global variables protected by semaphores and mutexes. Removed use of dataReaderThread. Add aca_thread_priority field to arrayent_config_t to allow application to configure the priority of ACA Gateway thread. Optimized stack utilization by defining messages as global variables. Add ArrayentSleep() and ArrayentWake() to allow application to disable/enable ACA daemon. Increased the sleep time in AicdThread() to 200 ms. Reduced the DNS lookup frequency to (1/600) if 100 DNS lookup fails in a row.

- 1.3.0.0: 2014/02/14 Add multi-attribute message support. Add ArrayentSetConfigDefaults(), so apps may upgrade ACA without changing code. Require AES encryption. Fix ArrayentNetStatus() advertising cloud connectivity prematurely. Change arguments of ArrayentSendData() and ArrayentRecvData().

- 1.2.0.0: 2014/01/27 Add encryption support. Change ACA threads' priorities to 7. Automatically call ArrayentReset() after successful post-initialization call to ArrayentConfigure(). Fix ArrayentConfigure() accepting input that could later be rejected by ArrayentInit(). Fix main ACA thread not sleeping between loops on WICED.
- 1.1.2.0: 2014/01/09 Fix ArrayentRecvProperty() not setting its length argument to zero on receive failure. Add IAR ARM-compatible versions of the library to release packages.
- 1.1.1.0: 2013/12/16 Make ArrayentConfigure() take a monolithic configuration structure. Null-terminate properties received using ArrayentRecvProperty(). Rename library interface file to aca.h.
- 1.1.0.1: 2013/11 First public beta release.

1.3 Details

Your connected device firmware will be composed of two parts: an embedded client application and the Arrayent Connect Agent. The embedded application refers to the embedded software that drives your product. This part of the software is developed by your product engineering team. Your product engineering team will integrate the Arrayent Connect Agent and the embedded client application. This guide explains the functions available in the Arrayent Connect Agent Application Programming Interface (the Arrayent API) for communicating with the Arrayent cloud.

Arrayent will supply the Arrayent Connect Agent and a board support package to abstract the Arrayent Agent from the hardware-specific code. Therefore, it is important to identify the microcontroller hardware and development toolchain early in the development process, so that Arrayent can ensure that your hardware is supported. The Arrayent Connect Agent requires 32 to 64 KB of program memory and approximately 15KB of RAM on a 32-bit embedded processor.

Arrayent Connect Agent's duties include: Logging in to the Arrayent Cloud and managing your device's session with the Arrayent cloud. Accepting messages from your Embedded Client Application and forwarding these messages into the Arrayent cloud, and vice versa.

Messages are represented as key-value pairs. The key represents an attribute of your device, and the value represents the current value of that attribute. For example, suppose that your device has an embedded temperature sensor, which your device samples at a predefined interval (say every 10 minutes). To send this data into the Arrayent Cloud, your client application sends the following message to the Arrayent agent: "temperature 84"

The Arrayent Endpoint simply forwards the message up into the cloud.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

arrayent_config_t
Structure to use with `ArrayentConfigure()` to configure the ACA 7

arrayent_net_status_t
Arrayent network status - returned by `ArrayentNetStatus()` 8

arrayent_timestamp_t
Arrayent timestamp data type (output of `ArrayentGetTime()`) 8

Chapter 3

File Index

3.1 File List

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

external/arrayent/source/include/aca.h
Arrayent Connect Agent (ACA) library interface 11

Chapter 4

Data Structure Documentation

4.1 arrayent_config_t Struct Reference

Structure to use with `ArrayentConfigure()` to configure the ACA.

```
#include <aca.h>
```

Data Fields

- uint16_t **product_id**
- const char ∗ **product_aes_key**
- const char ∗ **load_balancer_domain_names** [3]
- uint16_t **load_balancer_udp_port**
- uint16_t **load_balancer_tcp_port**
- const char ∗ **device_name**
- const char ∗ **device_password**
- const char ∗ **device_aes_key**
- uint8_t device_can_multi_attribute:1
- uint8_t enable_logging:1

  set to 1 to allow ACC remote enable of logging output
- uint8_t aca_thread_priority
- uint16_t aca_stack_size

  Indicates the stack size for AICD thread.

4.1.1 Field Documentation

4.1.1.1 uint8_t arrayent_config_t::device_can_multi_attribute

Indicates that the device is capable of receiving multi-attribute messages. Setting this to 1 will cause the server to encapsulate all attributes sent to this device in multi- attribute messages. Currently unused.

4.1.1.2 uint8_t arrayent_config_t::aca_thread_priority

Indicates the thread priority at which AICD thread will be created.

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

- external/arrayent/source/include/aca.h

4.2 arrayent_net_status_t Struct Reference

Arrayent network status - returned by `ArrayentNetStatus()`

`#include <aca.h>`

Data Fields

- uint8_t server_ip_obtained:1

  Arrayent server IP has been found.
- uint8_t heartbeats_ok:1

  Heartbeats to the Arrayent server are fine.
- uint8_t using_udp:1

  Connection to Arrayent server is over UDP.
- uint8_t using_tcp:1

  Connection to Arrayent server is over TCP.
- uint8_t connected_to_server:1

  Connection to Arrayent server is okay.
- uint8_t product_key_ok:1

  Product key is Okay.
- uint8_t product_key_ok_username_bad:1

  Product key is Okay but Username is Invalid.
- uint8_t login_successful:1

  Log-in successfully to Arrayent server.
- uint8_t key_exchange_successful:1

  Key exchange to Arrayent server successfully.
- uint16_t **padding**:7
- uint32_t device_id

  Contains AMP-ID.
- uint32_t service_endpoint_id

  Contains service end point deviceID.

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

- external/arrayent/source/include/aca.h

4.3 arrayent_timestamp_t Struct Reference

Arrayent timestamp data type (output of `ArrayentGetTime()`)

`#include <aca.h>`

Data Fields

- uint16_t year

  year AD
- uint16_t month

  0-11 (0 = Jan, 1 = Feb...)
- uint16_t day

  1-31
- uint16_t hour

0-23

- uint16_t minute

  0-59

- uint16_t second

  0-59

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

- external/arrayent/source/include/aca.h

Chapter 5

File Documentation

5.1 external/arrayent/source/include/aca.h File Reference

Arrayent Connect Agent (ACA) library interface.

5.1.1 Detailed Description

Copyright (c) 2014 Arrayent Inc. Company confidential. Please contact `sales@arrayent.com` to get permission to use in your application.

5.1.2 Function Documentation

5.1.2.1 arrayent_return_e ArrayentSetConfigDefaults (arrayent_config_t ∗ *config*)

Populate an ACA configuration structure with default settings.

Use this function to populate an ACA configuration structure with default values. This futureproofs your application by allowing you to upgrade your version of ACA without changing any of your application code, even when the ACA configuration structure adds new features.

Note that your application must still set most fields in the ACA configuration structure after having called this function on it. This function only populates settings for optional features of ACA.

Parameters

config[out]	ACA configuration structure to populate.

Returns

Whether or not ACA default settings were placed into `config`. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.2 arrayent_return_e ArrayentConfigure (arrayent_config_t ∗ *config*)

Configure the Arrayent Connect Agent.

Use this function to set all of your device's Arrayent attributes before starting the ACA.

This function must be successfully called before calling `ArrayentConfigure()`.

If this function is called after `ArrayentInit()`, it will automatically call `ArrayentReset()` if the reconfiguration structure was accepted. In the unlikely case that reconfiguration is successful but resetting is not, the host application must retry resetting the ACA until it succeeds before attempting any communication through the ACA.

Parameters

`in`	*config*	Structure fully populated with Arrayent configuration arguments.

Returns

Indicates whether or not the configuration was accepted. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.3 arrayent_return_e ArrayentInit (void)

Start the ACA.

This function prepares the ACA for communication with the Arrayent server. Do not call this function until every Arrayent property has been configured successfully using `ArrayentConfigure()`. Doing so may cause the Arrayent thread to hang.

Note that before calling this function, the Arrayent code expects the user application to have seeded the machine's random number generator using srand().

Returns

Indicates if initialization was completed successfully. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.4 arrayent_return_e ArrayentHello (void)

Check connection to the ACA.

This function sends a hello message to the ACA. If the ACA interface is functioning correctly, the hello message will receive a response message from the routing daemon to indicate a working connection.

Returns

Indicates if the hello was verified. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_-return_e"

5.1.2.5 arrayent_return_e ArrayentNetStatus (arrayent_net_status_t ∗ *status*)

Check Arrayent network status.

This function returns a bit field indicating the network status of the ACA.

Parameters

`out`	*status*	Pointer to the status structure to be updated. See `arrayent_net_-status_t`.

Returns

Indicates if the status was updated. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_-return_e"

5.1.2.6 arrayent_return_e ArrayentSleep (arrayent_sleep_level_e *sleep_level*)

Put the ACA to sleep.

This function instructs the ACA to go to sleep until ArrayentWake() is called. In this state, the application can call ArrayentConfigure() to change the configuration.

Parameters

`in`	*sleep_level*	Sets the exact sleep behavior. ARRAYENT_DEEPSLEEP: Turns client connection state machine off. Sockets are closed and no messages are sent to the cloud.

Returns

Indicates whether or not the ACA was successfully suspended. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.7 arrayent_return_e ArrayentWake (void)

Wake up the ACA.

This function brings the routing daemon out of sleep, causing it to reestablish its connection with the Arrayent server.

Applications should poll ArrayentNetStatus() for success after this function has been called and before sending property updates to the cloud.

Returns

Indicates whether or not the ACA is now in the process of resuming. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.8 arrayent_return_e ArrayentReset (void)

Reset the ACA.

This function forces the ACA's state machine to reset and log back in to the server. It may be called at any time after initialization is complete.

Be sure to poll `ArrayentNetStatus()` for server connection before attempting Arrayent communication after a reset.

Returns

Indicates if the reset was accepted. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_-return_e"

5.1.2.9 arrayent_return_e ArrayentGetTime (arrayent_timestamp_t ∗ *timestamp*, int16_t *timezone*)

Retrieve the current time from the server.

This function blocks for up to three seconds waiting for a response from the server.

The month field of the returned timestamp is zero-based: January is represented by a 0, February is represented by a 1, and so on.

Parameters

`out`	*timestamp*	Pointer to structure to fill with current time
`in`	*timezone*	GMT offset of desired time, in hours

Returns

Indicates if the timestamp was received. If not ARRAYENT_SUCCESS, ∗timestamp does NOT contain a valid timestamp. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.10 arrayent_return_e ArrayentSetProperty (char ∗ *property*, char ∗ *value*)

Write a new property to the server.

This function sets an arbitrary property on the Arrayent servers.

Parameters

`in`	*property*	Pointer to the property name.

The month field of the returned timestamp is zero-based: January is represented by a 0, February is represented by a 1, and so on.

Parameters

`in`	*value*	Pointer to the property value.

Returns

Indicates if the property was set correctly. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.11 arrayent_return_e ArrayentRecvProperty (char ∗ *data*, uint16_t ∗ *len*, uint32_t *timeout*)

Receive a property message.

This function listens for a property message from the Arrayent cloud. It is blocking; it will not return until either a property message has been received from the server or the `timeout` argument has elapsed.

Parameters

`out`	*data*	Where to place the received property message. This buffer is not touched if no message is received.
`in,out`	*len*	Input: the length of the receive buffer. This buffer should be at least one character longer than the longest expected message, to allow for null-termination. Output: if a property message was received, this number is set to the length of the data written to the data buffer. In the event of an insufficiently sized receive buffer, the received message is truncated to the length of the passed buffer. The value of len does not include the null terminator, which is always set by this function. If no property message was received, this is set to zero.
`in`	*timeout*	Timeout for the receive operation to complete, in milliseconds. Use 0 to indicate no wait.

Returns

Indicates if a property was received from the server. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.12 arrayent_return_e ArrayentSetMultiAttribute (uint8_t ∗ *data*, uint16_t *len*)

Write multiple attributes to the server simultaneously.

This function sends a multi-attribute message to the Arrayent servers.

Parameters

in	*data*	Property buffer. See documentation on Arrayent multi-attribute message format for description of contents.
in	*len*	Length of property buffer. Currently limited by build time macro MAX_MESSA-GE_SIZE, default is about 128 bytes.

Returns

Indicates if the attributes were sent to and accepted by the server. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.13 arrayent_return_e ArrayentRecvMultiAttribute (uint8_t ∗ *data,* uint16_t ∗ *len,* uint32_t *timeout*)

Receive a multi-attribute message.

This function listens for a multi-attribute message from the Arrayent cloud. It is blocking; it will not return until either a multi-attribute message has been received from the server or the `timeout` argument has elapsed.

In order to use this call, you must have specified that your device is capable of receiving multi-attribute messages in your call to `ArrayentConfigure()`.

Parameters

out	*data*	Where to place the received multi-attribute message. This buffer is not touched if no message is received.
in,out	*len*	Input: the length of the receive buffer. This buffer should be at least as long as the longest expected message. Output: if a multi-attribute message was received, this number is set to the length of the data written to the data buffer. In the event of an insufficiently sized receive buffer, this value will equal the size of the buffer, and the received message is truncated to the size of the buffer. If no property message was received, this is set to zero.
in	*timeout*	Timeout for the receive operation to complete, in milliseconds. Use 0 to indicate no wait.

Returns

Indicates if a multi-attribute message was received from the server. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.14 arrayent_return_e ArrayentSendData (uint8_t ∗ *data,* uint16_t *data_type,* uint16_t *len*)

Send binary data with optional timeout. this function is maintained for legacy applications only.

Note that ArrayentSendData() is used only for custom parser applications. It is used for sending proprietary binary data to the Arrayent cloud. User must first make explicit arrangements with Arrayent Customer Support to use this, otherwise the function will not work as expected.

This function sends data messages to the Arrayent servers synchronously. It is blocking, and will not return until a response message has been received from the server or the timeout value expires.

Parameters

in	*data*	Pointer to the data for transmission.
in	*data_type*	Data type of data to send
in	*len*	Length of data to send

Returns

Indicates if the data was sent to the server correctly. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.15 arrayent_return_e ArrayentRecvData (uint8_t ∗ *data,* uint16_t ∗ *data_type,* uint16_t ∗ *len,* uint32_t *timeout*)

Receive a raw data message. this function is maintained for legacy applications only.

Note that ArrayentRecvData() is used only for custom parser applications. It is used for receiving proprietary binary data from the Arrayent cloud. User must first make explicit arrangements with Arrayent Customer Support to use this, otherwise the function will not work as expected.

This function attempts to receive a data message from the Arrayent cloud. It is blocking; it will not return until either a data message has been received from the server or the `timeout` argument has elapsed.

Parameters

`out`	*data*	Where to place the received data message. This buffer is not touched if no message is received.
`out`	*data_type*	Where to place the received data message sub-type. This buffer is not touched if no message is received.
`in,out`	*len*	Input: the length of the receive buffer. This buffer should be at least the length of the longest expected message. Output: if a data message was received, this number is set to the length of the data written to the data buffer. In the event of an insufficiently sized receive buffer, this value will not equal the length of the received data. If no data message was received, this is set to zero.
`in`	*timeout*	Timeout for the receive operation to complete, in milliseconds. Use 0 to indicate no wait.

Returns

Indicates if a data message was received from the server. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.16 arrayent_return_e ArrayentFactoryResetDevice (void)

Send factory reset the command.

This function is used for reset device to factory settings.

Returns

Indicates if the factory reset was accepted. On Success: ARRAYENT_SUCCESS On Failure: Error Code "arrayent_return_e"

5.1.2.17 arrayent_return_e ArrayentDebug (char ∗ *buf*)

Arrayent Debug

wrapper function for logging module.

Parameters

`in`	*ip*	address for sysLogConfiguration.

Returns

On Success: ARRYENT_SUCCESS On Failure: Error COde "arrayent_return_e"

5.1.3 Enumeration Type Documentation

5.1.3.1 enum arrayent_return_e

Enumerator

ARRAYENT_SUCCESS	Command completed successfully.
ARRAYENT_FAILURE	Command failed.
ARRAYENT_TIMEOUT	Command failed: timed out.
ARRAYENT_BUFFER_TOO_BIG	Command failed; buffer too large.
ARRAYENT_CANNOT_MULTI_PROPERTY	Command failed; device is not configured for receiving multi-property messages.
ARRAYENT_ACCEPTED_CONFIG_BUT_CANNOT_RESET	Reconfiguration successful, but unable to reset afterward.
ARRAYENT_BAD_PRODUCT_ID	Product ID is invalid.
ARRAYENT_BAD_PRODUCT_AES_KEY	Product AES key is invalid.
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN_NAME_1	First load balancer is invalid.
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN_NAME_2	Second load balancer is invalid.
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN_NAME_3	Third load balancer is invalid.
ARRAYENT_BAD_LOAD_BALANCER_UDP_PORT	Load balancer UDP port is invalid.
ARRAYENT_BAD_LOAD_BALANCER_TCP_PORT	Load balancer TCP port is invalid.
ARRAYENT_BAD_DEVICE_NAME	Device name is invalid.
ARRAYENT_BAD_DEVICE_PASSWORD	Device password is invalid.
ARRAYENT_BAD_DEVICE_AES_KEY	Device AES key is invalid.
ARRAYENT_BAD_ACA_THREAD_PRIORITY	Device ACA thread priority is invalid.
ARRAYENT_BAD_ACA_STACK_SIZE	Device ACA stack size is invalid.
ARRAYENT_BAD_CONFIGS	Device is not configured.
ARRAYENT_FAIL_LOCK_MUTEX_RX_CTRL	Failed to get mutex lock for mutexRxCtrl.
ARRAYENT_FAIL_LOCK_MUTEX_TX_CTRL	Failed to get mutex lock for mutexTxCtrl.
ARRAYENT_FAIL_LOCK_MUTEX_TX_PROP	Failed to get mutex lock for mutexTxProp.
ARRAYENT_FAIL_LOCK_MUTEX_RX_PROP	Failed to get mutex lock for mutexRxProp.
ARRAYENT_FAIL_PUT_SEM_TX_CTRL	Failed to put semaphore for semTxReq.
ARRAYENT_FAIL_PUT_SEM_TX_PROP	Failed to put semaphore for semTxReq.
ARRAYENT_MSG_DATA_TOO_LONG	Data message size exceeds.
ARRAYENT_RX_CTRL_RSP_STATUS_FAILURE	mRxRsp response status is failure
ARRAYENT_BAD_MSG_TYPE	Invalid msg type.
ARRAYENT_CREATE_GATEWAY_THREAD_FAIL	Arrayent fail to create gateway thread.
ARRAYENT_DEAMON_NOT_INITIALIZED	Arrayent deamon is not initialized.
ARRAYENT_INVALID_INPUT_ARGUMENT	Arrayent invalid input arguments.
ARRAYENT_NO_GATEWAY_SESSION_EXIST	Gateway session does not exist.
ARRAYENT_INVALID_RESPONSE	Old response received.

Index

ARRAYENT_ACCEPTED_CONFIG_BUT_CANNOT_-RESET
aca.h, 17
ARRAYENT_BAD_ACA_STACK_SIZE
aca.h, 17
ARRAYENT_BAD_ACA_THREAD_PRIORITY
aca.h, 17
ARRAYENT_BAD_CONFIGS
aca.h, 17
ARRAYENT_BAD_DEVICE_AES_KEY
aca.h, 17
ARRAYENT_BAD_DEVICE_NAME
aca.h, 17
ARRAYENT_BAD_DEVICE_PASSWORD
aca.h, 17
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN_NA-ME_1
aca.h, 17
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN_NA-ME_2
aca.h, 17
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN_NA-ME_3
aca.h, 17
ARRAYENT_BAD_LOAD_BALANCER_TCP_PORT
aca.h, 17
ARRAYENT_BAD_LOAD_BALANCER_UDP_PORT
aca.h, 17
ARRAYENT_BAD_MSG_TYPE
aca.h, 17
ARRAYENT_BAD_PRODUCT_AES_KEY
aca.h, 17
ARRAYENT_BAD_PRODUCT_ID
aca.h, 17
ARRAYENT_BUFFER_TOO_BIG
aca.h, 17
ARRAYENT_CANNOT_MULTI_PROPERTY
aca.h, 17
ARRAYENT_CREATE_GATEWAY_THREAD_FAIL
aca.h, 17
ARRAYENT_DEAMON_NOT_INITIALIZED
aca.h, 17
ARRAYENT_FAIL_LOCK_MUTEX_RX_CTRL
aca.h, 17
ARRAYENT_FAIL_LOCK_MUTEX_RX_PROP
aca.h, 17
ARRAYENT_FAIL_LOCK_MUTEX_TX_CTRL
aca.h, 17
ARRAYENT_FAIL_LOCK_MUTEX_TX_PROP
aca.h, 17
ARRAYENT_FAIL_PUT_SEM_TX_CTRL
aca.h, 17
ARRAYENT_FAIL_PUT_SEM_TX_PROP
aca.h, 17
ARRAYENT_FAILURE
aca.h, 17
ARRAYENT_INVALID_INPUT_ARGUMENT
aca.h, 17
ARRAYENT_INVALID_RESPONSE
aca.h, 17
ARRAYENT_MSG_DATA_TOO_LONG
aca.h, 17
ARRAYENT_NO_GATEWAY_SESSION_EXIST
aca.h, 17
ARRAYENT_RX_CTRL_RSP_STATUS_FAILURE
aca.h, 17
ARRAYENT_SUCCESS
aca.h, 17
ARRAYENT_TIMEOUT
aca.h, 17
aca.h
ARRAYENT_ACCEPTED_CONFIG_BUT_CANN-OT_RESET, 17
ARRAYENT_BAD_ACA_STACK_SIZE, 17
ARRAYENT_BAD_ACA_THREAD_PRIORITY, 17
ARRAYENT_BAD_CONFIGS, 17
ARRAYENT_BAD_DEVICE_AES_KEY, 17
ARRAYENT_BAD_DEVICE_NAME, 17
ARRAYENT_BAD_DEVICE_PASSWORD, 17
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN-_NAME_1, 17
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN-_NAME_2, 17
ARRAYENT_BAD_LOAD_BALANCER_DOMAIN-_NAME_3, 17
ARRAYENT_BAD_LOAD_BALANCER_TCP_PO-RT, 17
ARRAYENT_BAD_LOAD_BALANCER_UDP_PO-RT, 17
ARRAYENT_BAD_MSG_TYPE, 17
ARRAYENT_BAD_PRODUCT_AES_KEY, 17
ARRAYENT_BAD_PRODUCT_ID, 17
ARRAYENT_BUFFER_TOO_BIG, 17
ARRAYENT_CANNOT_MULTI_PROPERTY, 17
ARRAYENT_CREATE_GATEWAY_THREAD_FA-IL, 17
ARRAYENT_DEAMON_NOT_INITIALIZED, 17
ARRAYENT_FAIL_LOCK_MUTEX_RX_CTRL, 17

ARRAYENT_FAIL_LOCK_MUTEX_RX_PROP, 17
ARRAYENT_FAIL_LOCK_MUTEX_TX_CTRL, 17
ARRAYENT_FAIL_LOCK_MUTEX_TX_PROP, 17
ARRAYENT_FAIL_PUT_SEM_TX_CTRL, 17
ARRAYENT_FAIL_PUT_SEM_TX_PROP, 17
ARRAYENT_FAILURE, 17
ARRAYENT_INVALID_INPUT_ARGUMENT, 17
ARRAYENT_INVALID_RESPONSE, 17
ARRAYENT_MSG_DATA_TOO_LONG, 17
ARRAYENT_NO_GATEWAY_SESSION_EXIST, 17
ARRAYENT_RX_CTRL_RSP_STATUS_FAILURE, 17
ARRAYENT_SUCCESS, 17
ARRAYENT_TIMEOUT, 17

aca.h
- arrayent_return_e, 17
- ArrayentConfigure, 11
- ArrayentDebug, 16
- ArrayentFactoryResetDevice, 16
- ArrayentGetTime, 13
- ArrayentHello, 12
- ArrayentInit, 12
- ArrayentNetStatus, 12
- ArrayentRecvData, 16
- ArrayentRecvMultiAttribute, 15
- ArrayentRecvProperty, 14
- ArrayentReset, 13
- ArrayentSendData, 15
- ArrayentSetConfigDefaults, 11
- ArrayentSetMultiAttribute, 14
- ArrayentSetProperty, 14
- ArrayentSleep, 12
- ArrayentWake, 13

aca_thread_priority
- arrayent_config_t, 7

arrayent_config_t, 7
- aca_thread_priority, 7
- device_can_multi_attribute, 7

arrayent_net_status_t, 8

arrayent_return_e
- aca.h, 17

arrayent_timestamp_t, 8

ArrayentConfigure
- aca.h, 11

ArrayentDebug
- aca.h, 16

ArrayentFactoryResetDevice
- aca.h, 16

ArrayentGetTime
- aca.h, 13

ArrayentHello
- aca.h, 12

ArrayentInit
- aca.h, 12

ArrayentNetStatus
- aca.h, 12

ArrayentRecvData
- aca.h, 16

ArrayentRecvMultiAttribute
- aca.h, 15

ArrayentRecvProperty
- aca.h, 14

ArrayentReset
- aca.h, 13

ArrayentSendData
- aca.h, 15

ArrayentSetConfigDefaults
- aca.h, 11

ArrayentSetMultiAttribute
- aca.h, 14

ArrayentSetProperty
- aca.h, 14

ArrayentSleep
- aca.h, 12

ArrayentWake
- aca.h, 13

device_can_multi_attribute
- arrayent_config_t, 7

external/arrayent/source/include/aca.h, 11