Meru Networks Location Feed

Reference Guide

Meru Networks Location Feed Reference Guide

July 2014

Location Feed

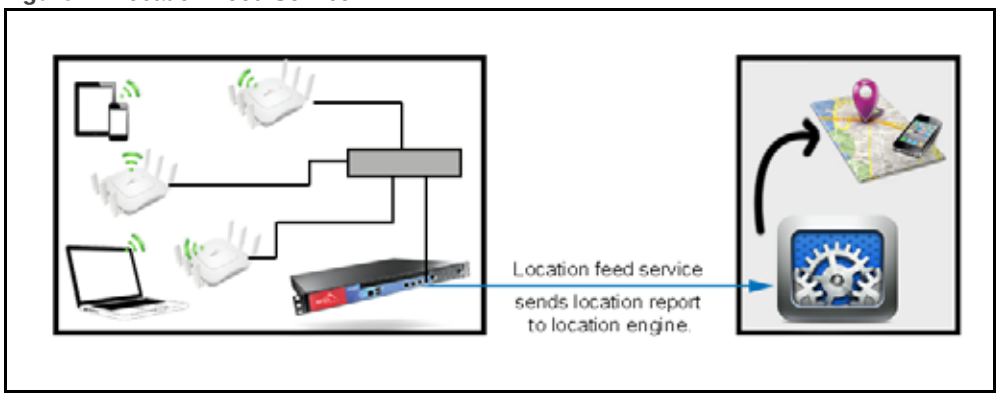
The location feed service captures parameters that your location engine can use to locate the position of a client / station in your network.

The location feed service is currently supported only for clients/stations connected to AP832 and AP822.

How It Works

By default, the location feed service is disabled. When enabled, the location feed service captures various parameters (Location Report) at pre-defined intervals (default 3 seconds) and sends them as UDP packets to your location engine. You can use the location feed SDK to create a customized interface (wrapper) around your location engine to capture and process data from Meru controller location feed service.

Figure 1: *Location Feed Service*



Using Location Feed Service

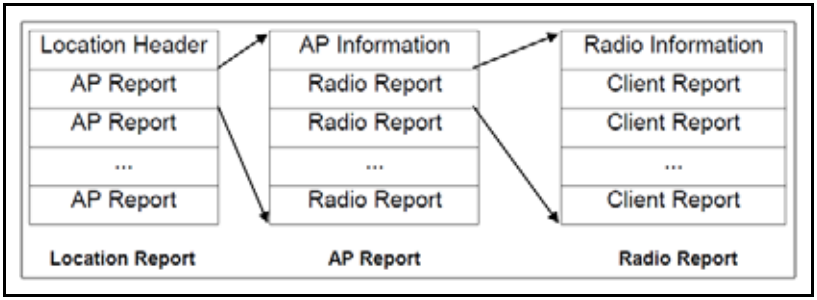
To allow the controller to start sending location reports, enable the location feed service, specify the IP address and the port of your location engine and configure the time interval at which the location feed data is sent to your location engine (**LE**).

```
default(15)# configure terminal

default(15)(config)# location-server enable

default(15)(config)# location-server ip-address <ip-address of LE>

default(15)(config)# location-server port <port of LE>

default(15)(config)# location-server report-interval <report interval in sec-
   onds>
```

Location Report

A location report (which is sent as a UDP packet) is an encapsulation of one or more AP Report that further contains one or more AP radio report which contains one or more client reports.

Figure 2: *Location Feed Responses*



Typically the location report message format (packet) contains the following

- Version: Version of the location feed
- Response (Message) Type: Message type
- Header Length: Specifies if the response contains additional headers.
- Payload Length: Location feed payload length

Location Feed Message (Response) Types

The location feed SDK has a list of API that you can use to request and parse location feed data. Request from your location feed engine are responded using the following message types:

TABLE 1: *Location Feed Response Type and Description*

Response Message	Description	Header Length and Description
LOC_MSG_TYPE_DATA	The data part of location containing AP report, Radio report and Client report.	No additional data is available as part of this response.

TABLE 1: *Location Feed Response Type and Description*

Response Message	Description	Header Length and Description
`LOC_MSG_-TYPE_GET_AP_REQ`	This message type is used to get the list of all AP and radio info from controller. The location engine should send this request to controller on port 4301.	1 – Signifies control message sent by the location engine contains additional header. For this message, additional header contains the Request ID that is set by the location engine.
`LOC_MSG_-TYPE_GET_AP_RESP`	This message type is used to send the list of all AP and radio info from controller. This message is sent in response to `LOC_MSG_-TYPE_GET_AP_REQ message` received by controller from your location engine.	2- Signifies control message sent by the location feed service in the controller to your location engine contains the following additional data: **Response ID**: This is set to the requestID sent by your location engine to the location feed service. **Sequence ID**: If the AP information cannot be sent in a single packet, it is split across sequence of packets and this value represents the part of sequence. **Last In Sequence**: If the AP information is sent in parts then this flag is set to denote the last packet in the sequence.

AP Report

AP report contains MAC address of the Ethernet interface of the AP and an optional TLV value that specifies if the AP is up or down. The following are captured as part of an AP report

AP MAC: The MAC address of the Ethernet interface of the AP

Radio Report

Radio report contains details about the radio channel in use, the channel width and the radio mode (a/b/g/n).

Client Report

Client report contains the client MAC address, RSSI value, and flag that specify the clients connection status.

TABLE 2: *CLIENT REPORT FLAG DETAILS*

Flag	Description
0	Client is not connected to the AP
1	Client is discovered but not associated to AP
2	Client is assigned to an AP

Location Feed SDK

The location feed SDK is a set of C libraries that provide interfaces to capture responses from the controllers location feed service and parse them appropriately to your location engine.

parse_loc_header

Used to parse the location header on receiving the packet from the controller.

Usage: `loc_feed_t parse_loc_header (void *loc_buf);`

- Parameters: loc_buf
- Return Value: Returns the pointer to location header or NULL on error

get_total_ap_records

Used to get the total number of AP records in the location feed message from the controller.

Usage: `int get_total_ap_records (loc_feed_t *loc_feed);`

- Parameters: loc_feed (Pointer to location header)
- Return Value: An integer value if there are total AP records in feed, else **-1** on error.

get_first_ap_record

Used to get the first AP record in the location feed.

Usage: `ap_record_t *get_first_ap_record (loc_feed_t *loc_feed)`

- Parameters: loc_feed: Pointer to location header.
- Return Value: Pointer to first AP report and NULL if none.

get_next_ap_record

Used to get the next AP record given the current AP record.

Usage: `ap_record_t *get_next_ap_record (ap_record_t *cur_ap_rec);`

- Parameters:
  - cur_ap_rec—Pointer to current AP record
- Return Value: Pointer to next AP record on NULL if current is last

get_next_ap_rec_tlv

Used to process each TLV in the ap record. The first cur_tlv is passed as NULL.

Usage: `ap_tlv_t *get_next_ap_rec_tlv (ap_record_t *ap_rec, ap_tlv_t *cur_tlv);`

- Parameters:
  - oap_rec—AP records for which TLV parsing is needed.
  - ocur_tlv—current TLV pointer. For the first TLV parsing this is NULL.
- Return Value: Pointer to TLV or NULL if there is no more TLVs

get_total_radio_records

Used to get the total number of radio records in the location feed for the given AP record.

Usage: `int get_total_radio_records (ap_record_t *ap_rec);`

- Parameters:
  - ap_rec—Pointer to AP record.
- Return Value: Total radio records in AP record else -1 on error.

get_first_radio_record

Used to get the first radio record for a given AP record.

Usage: `radio_record_t *get_first_radio_record (ap_record_t ap_rec);`

- Parameters:
  - ap_rec—Pointer to AP record
- Return Value: Pointer to first radio record in AP record and NULL if none.

get_next_radio_record

Used to get the next radio record given the current radio record.

Usage: `radio_record_t *get_next_radio_record (radio_record_t *cur_radio_rec);`

- Parameters:
  - cur_radio_rec—Pointer to current radio record.
- Return Value: Pointer to next radio record on NULL if current is last

get_next_radio_rec_tlv

Used to process each TLV in the radio record. First time cur_tlv is passed as NULL

Usage: `radio_tlv_t *get_next_radio_rec_tlv (radio_record_t *radio_rec, radio_tlv_t *cur_tlv);`

- Parameters:
  - oradio_rec—Radio records for which TLV parsing is needed.
  - ocur_tlv—current TLV pointer. For the first TLV parsing this is NULL.
- Return Value: Pointer to TLV or NULL if there are no more TLVs

get_total_client_records

Used to get the total number of client records in the location feed for the given radio record.

Usage: `int get_total_client_records (radio_record_t *radio_rec);`

- Parameters
  - radio_rec—Pointer to radio record
- Return Value: Total client records in radio record else -1 on error.

get_first_client_record

Used to get the first client record for a given radio record.

Usage: `client_record_t *get_first_client_record (radio_record_t radio_rec);`

- Parameters:
  - radio_rec—Pointer to radio record
- Return Value: Pointer to first client record in radio record and NULL if none.

get_next_client_record

Used to get the next client record given the current client record.

Usage: `client_record_t *get_next_client_record (client_record_t *cur_client_rec);`

- Parameters
  - cur_client_rec—Pointer to current client record
- Return Value : Pointer to next client record on NULL if current is last

get_next_client_rec_tlv

Used to process each TLV in the client record. First time cur_tlv is passed as NULL.

Usage: `client_tlv_t *get_next_client_rec_tlv (client_record_t *client_rec, client_tlv_t *cur_tlv);`

- Parameters:
  - oclient_rec—Client records for TLV that needs to be parsed.
  - ocur_tlv—Current TLV pointer. For the first TLV parsing this is NULL.
- Return Value : Pointer to TLV or NULL if there is no more TLVs

Sample Codes

The following code samples, written in C language, illustrate making location request calls using the APIs and followed by the code to parse response from the System Director location service.

Making a Request

```
{
    char buf[BUFLEN];
    int slen = sizeof(se_addr);
    loc_feed_t *loc_feed;
    ap_record_t *ap_rec;
    radio_record_t *rd_rec;
    client_record_t *cl_rec;
    client_tlv_t *cl_tlv;
    int ap_offset = 0;
    int cl_offset = 0;
```

```
   int len;
   loc_get_ap_req_hdr_t loc_req_hdr;

   /* Build a location get request message to get all APs */
   loc_req_hdr.request_id = 1;
   len = loc_build_feed_header(buf, LOC_VERSION, LOC_MSG_TYPE_GET_AP_REQ,
                               sizeof(loc_req_hdr), (void *)&loc_req_hdr);
   /* Socket for Location Engine control messages */
   if ((cl_sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
       printf("\nCannot create socket");
       return -1;
   }
   memset((char *)&cl_addr, 0, sizeof(cl_addr));
   cl_addr.sin_family=AF_INET;
   cl_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
   cl_addr.sin_port=htons(0x10cd);
   printf("\nSent Get all AP request to location daemon");

   /* Socket for Location feed messages from controller */
   if ((se_sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1) {
       printf("\nCannot create socket");
       return -1;
   }

   memset((char *)&se_addr, 0, sizeof(se_addr));
   se_addr.sin_family = AF_INET;
   se_addr.sin_port = htons(10000);       /* Default port of Location engine config-
ured on controller */
   se_addr.sin_addr.s_addr = htonl(INADDR_ANY);
   if (bind(se_sock, (struct sockaddr *)&se_addr, sizeof(se_addr)) == -1) {
       printf("\nCannot bind");
       return -1;
}
```

```
    /* Send Get ALL AP request */
    if (sendto(cl_sock, buf, len, 0, (struct sockaddr *)&cl_addr,
               sizeof(cl_addr)) == -1) {
        printf("\nError sending Get all AP request to location daemon %s\n",
               strerror(errno) );
        return 0;
 }
    /* Receive message which will contain one GET ALL APs and all location feed */
    while (1) {
        if (recvfrom(se_sock, buf, BUFLEN, 0, (struct sockaddr *)&se_addr,
                    &slen)==-1) {
            printf("\nError in packet recv");
 }
        /* Call an API which will invoke the apis to parse TLVs */
        process_location_buf(buf);
 }
```

Handling Response

```
int process_location_buf (void *loc_buf)
{
    loc_feed_t  *loc_feed = NULL;
    loc_get_ap_resp_hdr_t *loc_resp_hdr;
    loc_feed = parse_loc_header(loc_buf);
    printf("\n----------------------------------------------------------------");
    printf("\nLocation header ");
    printf("\n----------------------------------------------------------------");
    printf("\nLocation version          : %u"
           "\nLocation Msg type         : %u"
           "\nLocation Header Length    : %u"
           "\nLocation Payload Length   : %u",
          loc_feed->version, loc_feed->msg_type, loc_feed->header_len, loc_feed->pay-
 load_len);
    /* Get ALL AP response */
```

```
if (loc_feed->msg_type == LOC_MSG_TYPE_GET_AP_RESP) {
    loc_resp_hdr = (loc_get_ap_resp_hdr_t *)loc_feed->data;
    printf("\n-------------------------------------------------------------");
    printf("\nLocation Response Id      : %u"
           "\nLocation Sequence Id      : %u"
           "\nLocation Last sequence    : %s",
           loc_resp_hdr->response_id, loc_resp_hdr->seq_id,
           ((loc_resp_hdr->last_in_seq == 1) ? "Yes" : "No"));
}
```