Intel® Retail Client Manager Audience Analytics

Socket API Reference Manual

September 2014

Contents

Tables

Revision History

Rev.	Description	Date
-001	Initial release.	June 2014
-002	Added Prerequisite section.	September 2014

§

Prerequisites

Prior to using the Intel® RCM Audience Analytics Socket API, an Audience Analytics License must be allocated to the Intel RCM Player and the Intel RCM Player must be started.

§

1 Introduction

The Intel® Retail Client Manager Audience Analytics (Intel® RCM Audience Analytics) Socket API allows developers to connect to Intel® RCM Audience Analytics using TCP/IP. This allows for a platform-independent interface, with support for both local and remote communication between Intel® RCM Audience Analytics and third-party applications.

The API allows an application to communicate with Intel® RCM Audience Analytics in one of two ways:

- An application can request status information from Intel® RCM Audience Analytics (for example, the number of people viewing a display).
- An application can set up events or triggers based on certain audience attributes so that Intel® RCM Audience Analytics will issue event messages only when the desired conditions are met.

The target audience for this document is software developers that are familiar with programming TCP/IP sockets, and that require real-time access to the Intel® RCM Audience Analytics application.

1.1 Configuring the API server

To configure the API in Intel® RCM Audience Analytics software, follow these steps:

1. From within the Intel® RCM Audience Analytics Configurator, select *Main* > *Configure*.
2. In the dialog box, look for the section labeled *API Server Settings*.
3. Verify that the *Enable API Server* checkbox is checked.
4. Select the desired port number for the API server (default 12500).
5. Click *OK* to confirm the changes.

An application can then connect to Intel® RCM Audience Analytics based on the IP address or name of the Intel® RCM Audience Analytics platform and the configured port number. The port settings will be saved so that the servers can be restarted at the last ports automatically on startup.

§

2 Audience Counter API

Applications communicate with the Intel® RCM Audience Analytics Audience Counter API using a standard five-byte header, as shown in Table 2-1.

Table 2-1 MessageHeader structure

Field	Size	Value	Description
MagicWord	2 bytes	0xFACE	A simple code that can be used to validate that the message was received successfully.
Version	1 byte	0x01	The version of the API that is currently active.
Type	1 byte	Variable	The type of data (either a message command or message event described in Sections 2.1 and 2.2).
PayloadSize	1 byte	Variable	The size of the data that follows this header.

NOTE: All multibyte parameters are in network (big-endian) order.

Additional parameters can be specified by sending them immediately following the header (depending on the type of message being sent) and then adjusting the *PayloadSize* variable in the header appropriately. Note that the *PayloadSize* does not include the header size.

When the appropriate command has been successfully issued, the Intel® RCM Audience Analytics Audience Counter will send a response as a stream of bytes. The return value will be contained in the standard 5-byte message header along with any additional payload.

Two types of messages can be encapsulated into the standard message header:

- **Message commands.** These are issued by client applications to the API server.
- **Message events.** These are sent by the API server in response to Message Commands.

See *Appendix A Sample Java Code* on how to interface to the API.

2.1 Message commands

The following subsections present the current set of available Message Commands that a client application can issue to the Intel® RCM Audience Analytics Audience Counter. The appropriate command value is what will be set in the *Type* field of the *ViewerEvent* structure.

2.1.1 apiGetAudienceStatus

Command	0
Description	Requests immediate status information from the Intel RCM Audience Analytics Audience Counter regarding number of viewers currently in front of the display.
Payload	None. The payload size should be 0.
Response	On success, an EVENT_AUDIENCE_STATUS message will be sent. On failure, an EVENT_NACK message will be sent.

2.1.2 apiGetAudienceDetails

Command	1
Description	Requests immediate status information from the Intel RCM Audience Analytics Audience Counter regarding details of all viewers currently in front of the display.
Payload	None. The payload size should be 0.
Response	On success, an EVENT_AUDIENCE_DETAILS message will be sent. On failure, an EVENT_NACK message will be sent.

2.1.3 apiGetMajorityGender *DEPRECATED*

Command	2
Description	Requests the majority gender of audience members from the Intel RCM Audience Analytics Audience Counter over the past 15 seconds.
Payload	None. The payload size should be 0.
Response	On success, an EVENT_MAJORITY_GENDER message will be sent. On failure, an EVENT_NACK message will be sent.

Caution: This command has been deprecated. Do not use this command to determine instantaneous majority gender. Instead, use the *apiGetAudienceDetails* message.

2.1.4 apiGetViewerEvents

Command	5
Description	Requests continuous information from the Intel RCM Audience Analytics Audience Counter regarding number of viewers currently in front of the display. Events are sent each time a viewer starts or stops looking at the display, or when another significant visual change occurs (such as a viewer changing directions).
Payload	One byte representing either to start sending events (value 1) or to stop sending events (value 0).
Response	On success, an EVENT_ACK will be sent. On failure, an EVENT_NACK will be sent.

2.2 Message events

The following list summarizes the various types of Message Events that can be sent by the API server in response to commands. The appropriate message value is what will be set in the *Type* field of the *ViewerEvent* structure.

2.2.1 EVENT_ACK

Message	128
Description	Specifies a success code for a previously issued command.
Payload	0 bytes

2.2.2 EVENT_NACK

Message	129
Description	Specifies a failure or error code for a previously issued command.
Payload	0 bytes

2.2.3 EVENT_AUDIENCE_STATUS

Message	130
Description	Specifies the number of viewers currently in front of the display.
Payload	One byte representing the size of the audience.

2.2.4 EVENT_AUDIENCE_DETAILS

Message	131
Description	Returns a detailed list of all viewers (up to a maximum of 12 viewers) currently in front of the display.
Payload	One byte representing the number of viewers N, followed by N * 20 bytes containing N `ViewerDetail` data structures (see below).

2.2.5 EVENT_MAJORITY_GENDER

Message	132
Description	Returns the majority gender of the audience over the past 15 seconds.
Payload	One byte representing the majority gender, where a value of 0 is unknown, 1 is male, and 2 is female.

2.2.6 EVENT_VIEWER

Message	135
Description	Returns information for a viewer either when the viewer starts or stops viewing the display.
Payload	21 bytes consisting of a `ViewerEvent` data structure (see below).

§

3 Data Structures

This section describes the data structures used by the Intel® RCM Audience Analytics Socket API.

Table 3-1 ViewerDetail structure

Field	Size	Value	Description
ID	4 bytes	Variable	A unique unsigned integer that defines an individual (the ID values will reset to zero when the 32-bit limit is reached).
Gender	1 byte	Variable	Defines the gender of the individual. See Note 1.
Age	1 byte	Variable	Defines the age of an individual. See Note 2.
Reserved	1 byte	Variable	Reserved for future use.
Reserved 2	1 byte	Variable	Reserved for future use.
ViewingTime	4 bytes	Variable	Defines the number of milliseconds that an individual has been facing towards the camera.
Top-left X	2 bytes	Variable	The top-left *X* position of the rectangle around the individual, in unsigned 16-bit (0 to 65535) normalized coordinates. See Note 3.
Top-left Y	2 bytes	Variable	The top-left *Y* position of the rectangle around the individual, in unsigned 16-bit (0 to 65535) normalized coordinates. See Note 3.
Face Width	2 bytes	Variable	The width of the rectangle around the individual, in unsigned 16-bit normalized coordinates. See Note 3.
Face Height	2 bytes	Variable	The height of the rectangle around the individual, in unsigned 16-bit normalized coordinates. See Note 3.

NOTE:

1. For gender, a value of 0 represents an unknown gender, 1 represents a male, and 2 represents a female.
2. For age, a value of 0 represents an unknown age, 1 represents a child (under 16), 3 represents a young adult (16 to 34), 4 represents an older adult (35 to 64), and 5 represents a senior (65+). A value of 2 represents a teenager (13 to 19), but the software does not currently output this age class.
3. Normalization allows coordinates and dimensions of faces to be represented in a form that is independent of the resolution of the camera that a face was detected in. For example, an x coordinate of 32768 always represents the center horizontal location irrespective of the camera capture width. This could be converted into a pixel location in an image with dimensions *image_width* × *image_height* using the following formula: pixel_x = normalized_x * image_width / 65535.

Table 3-2 ViewerEvent structure

Field	Size	Value	Description
Type	1 byte	Variable	Represents the event type as an unsigned byte. See Note 1.
ViewerDetail	20 bytes	Variable	The ViewerDetail data structure

NOTE: For the event type, a value of 0 represents a “viewer start” event, and a value of 2 represents a “viewer end” event.

§

4 Additional Development Notes

4.1 Combining commands and events

If you combine viewer update events with polling commands, the order of the command response and event messages may vary. For example, if you send an *apiGetAudienceDetails* command while also receiving viewer events, it is possible to receive a subsequent "viewer start" event for a viewer that was already in the response message to the *apiGetAudienceDetails* command. Therefore, it is always important to track and verify the ID values of all *ViewerDetail* structures that your application may receive using the API.

4.2 Analytics applications

This API is best suited for interactive applications that need to make real-time decisions based on the audience, such as gender-based content-triggering. However, we do not recommend using it to gather viewership data for analytics purposes since it does not filter for false positives.

4.3 API integration best practices

The following list of integration best practices will instruct and guide users for optimal configuration and usage of the Intel® RCM Audience Analytics Socket API and the integration application.

- A feature should be provided to allow a user to easily verify that the integration software can successfully connect to Intel® RCM Audience Analytics. It should be possible to verify this connection when the software is first set up and any time after the software has fully been configured.
- The client should issue no more than four commands per second to the Intel® RCM Audience Analytics Socket API.
- It should be possible to specify content triggering rules that use only gender information, only age information, or both gender and age information.
- We do not recommend performing content triggering specifically for children, as this may violate federal, state, and municipal laws in some jurisdictions.
- It should be possible to specify content triggering rules that use audience information that is no more than two seconds old. This implies that a polling command, such as *apiGetAudienceDetails*, would be issued no more than two seconds prior to making a corresponding content-triggering decision.
- Use the *apiGetAudienceDetails* command to determine instantaneous majority gender or majority age. (Note that it is often a mistake to use the *apiGetMajorityGender* command for content triggering because this command does not provide the instantaneous majority gender, but rather an aggregated majority gender over the past 15 seconds.)
- Avoid using *EVENT_VIEWER* events (such as when a viewer starts looking) to make content-triggering decisions based on gender or age. It is better to rely on polling commands such as *apiGetAudienceDetails* because *EVENT_VIEWER* events for when a viewer starts looking will contain slightly less accurate gender and age information on average.

§

A Sample Java Code

The following Java* source code demonstrates how to send a command message to the Intel® RCM Audience Analytics Socket API and retrieve a response. The primary purpose of this code is to show how a packet can be assembled correctly, and how to extract data from the response packet.

```
import java.io.*;
import java.net.*;

public class ConnectTest
{
  public static final int MESSAGE_GET_AUDIENCE_STATUS = 0;
  public static final int EVENT_ACK = 128;
  public static final int EVENT_NACK = 129;
  public static final int EVENT_AUDIENCE_STATUS = 130;

  public static void main(String[] args)
  {
    try
    {
      System.out.println("Start RCM Audience Analytics Connect Test");

      // Open a socket on the local host, port 12500 (default for RCM Audience Analytics)
      Socket myClient;
      myClient = new Socket("localhost", 12500);

      System.out.println("myClient = " + myClient);

      DataInputStream input;
      DataOutputStream output;
      input = new DataInputStream(myClient.getInputStream());
      System.out.println("input = " + input);
      output = new DataOutputStream(myClient.getOutputStream());
      System.out.println("output = " + output);

      // First we will set up the request message as a byte stream
      byte request[] = new byte[5];
      request[0] = (byte)0xFA; // Magic word (first byte)
      request[1] = (byte)0xCE; // Magic word (second byte)
      request[2] = (byte)0x01; // Version
      request[3] = (byte)MESSAGE_GET_AUDIENCE_STATUS; // Command
      request[4] = (byte)0x00; // Payload size (0 for apiGetAudienceStatus)
      output.write(request, 0, 5); // Send the request

      boolean reading = true;
      int bytesRead = 0;

      // For apiGetAudienceStatus, we will read back
      // the 5 byte header + 1 byte for the payload.
      // Other response messages could have different
      // types of payload sizes, but this one is really simple.
      byte response[] = new byte[6];

      // Loop until we have read the desired number of bytes (or an error occurs)
      while(reading && bytesRead < 6)
      {
```

```
        try
        {
          response[bytesRead] = input.readByte();
          bytesRead++;
        }
        catch(IOException e)
        {
          reading = false;
        }
      }

      // Verify that the returned byte stream has the magic word
      if(response[0] == (byte)0xFA && response[1] == (byte)0xCE)
      {
        System.out.println("Magic word detected");

        // Verify that the version is correct
        if(response[2] == (byte)0x01)
        {
          System.out.println("Version is correct");

          // Verify that we got the desired response (EVENT_AUDIENCE_STATUS)
          if(response[3] == (byte)AIMViewConnectTest.EVENT_AUDIENCE_STATUS)
          {
            System.out.println("Received EVENT_AUDIENCE_STATUS message");

            // Verify that the payload size is correct
            if(response[4] == (byte)1)
            {
              // The audience size will be in the payload of the
              // returned message (located at element 5 in the array)
              System.out.println("Audience Size: " + response[5]);
            }
            else
            {
              System.out.println("Incorrect payload size: " + response[4]);
            }
          }
          else
          {
            System.out.println("Incorrect response message: " + response[3]);
          }
        }
        else
        {
          System.out.println("Incorrect version: " + response[2]);
        }
      }
      else
      {
        System.out.println("Incorrect magic word: 0x" +
          response[0] + response[1]);
      }

      input.close();
      output.close();
    }
    catch(Exception exp)
    {
      exp.printStackTrace();
    }
  }
}
```

§