MediaAUTH Draft Proposal

August 21, 2012

Contents

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [10].

1 Introduction

A lot of hybrid radio devices already require the user to login in order to use certain services proposed by the producer of the device (e.g. Frontier Silicon's Radio Portal [3] or Pure's The Lounge [8]). At the same time, hybrid radio services as RadioTAG benefit or require authentication to store user specific data in the broadcaster's systems. An individual account per broadcast channel is cumbersome and affects end-user experience severely.

MediaAUTH is a distributed system that allows broadcaster's and producer's services, the Service Providers (SPs), to authenticate users based on global accounts hosted by independent entities, the Authentication Providers (APs). A central directory is required to certify the authorization of SPs and APs.

Internally MediaAUTH uses OAuth 2.0 [4], with some minor additions to accommodate the main goals of its design:

- A unique user name and password per user for *all* stations and services.
- Storage of all user data at the service provider (i.e. the broadcaster). Broadcasters need to keep the control of their user data and store them on their own servers.
- Lightweight for the receiver / the authenticated device
- Based on existing standards (OAuth 2.0 [4], OAuth 2.0 bearer tokens [5]).

2 Service & User Perspective

This section describes the services provided by MediaAUTH and how the user interacts with them.

In principle, the user experiences the MediaAUTH system as being logged into the device he is using, while he is actually using different broadcaster's services such as RadioTAG.

2.1 Login

The user can log into the MediaAUTH system by supplying his e-mail address and his password to the device (see section 3.1). The device will then transparently handle authentication with individual broadcaster's services (see section 3.4). Some services that require authentication might also trigger the MediaAUTH login on the device. Being logged into the device corresponds to the paired scope of the RadioTAG specification [11].

2.2 Soft Login

If the user chooses not to log into the device immediately, the MediaAUTH system will create a temporary session for the user, identifying his actions (e.g. Tags) for later addition to his account (see section 3.1). At a later point, upon login or account creation, the system will re-associate the already existing data to the registered account (see section 3.3). Using a temporary identifier corresponds to the unpaired scope of the RadioTAG specification [11].

2.3 Account Creation

A user may choose to create an account directly on the device. Some devices already require this at the moment of purchase. To create an account, the user enters his e-mail address and a password. After confirmation of the e-mail address (i.e. clicking on a link in a special e-mail), the user can login to the MediaAUTH system using his e-mail and password on any MediaAUTH-enabled device (see section 3.2).

3 MediaAUTH Flows

This section describes on a higher level, how the different parts of the MediaAUTH infrastructure interact to provide the described services. Technical details can be found in section 4.

The descriptions use following roles:

Authentication Provider The AP is responsible for identifying and authenticating users. A user may have a registered user account or may be issued a temporary identifier (unpaired scope). In the MediaAUTH system, there exist multiple APs, but only a single authoritative AP per user (registered or temporary).

Directory The central Directory is responsible for the authorization of APs and SPs to use the MediaAUTH system. Further, it stores which AP is authoritative for a given user (registered users only).

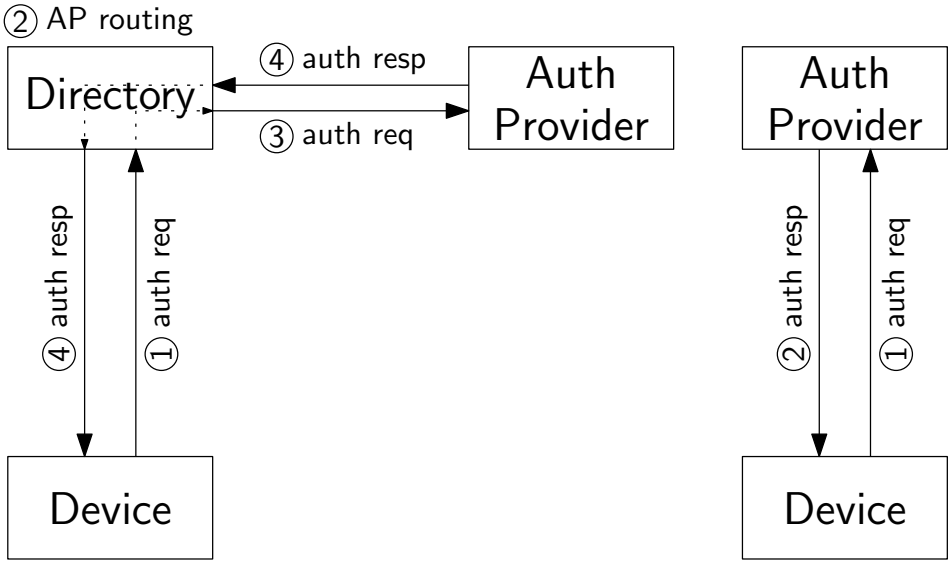
Device The device that receives audio/video broadcasts and through which additional services are provided to the user.

Service Provider An SP is any entity using the MediaAUTH system as a service to provide a particular service to the device or the user.

3.1 AP Login

This flow is used in two cases:

- A registered user enters his user name and password to associate with its AP.
- A device decides to establish a temporary user token (e.g. for unpaired tagging).



(a) Routed Flow (b) Simple Flow

Figure 1: AP Login Flows for MediaAUTH

Further, there exist two versions of the flow: Routed AP login, used in the case where a device does not know the responsible AP for the user (or for a temporary token) and simplified login, when the device knows the responsible AP from a previous login. Both flows are depicted in figure 1.

In the following, the steps of the routed login flow are explained in detail.

1. **Authentication Request** The device requests authentication from its AP through the Directory by initializing a Resource Owner Password Credentials Grant [4, section 4.3.] (in the case of a registered user) or a simple request for a token (in the case of a temporary user token request).

2. **AP Routing** The Directory forwards the request to the responsible AP (based on the user name in case of a registered user, to any AP in case of a temporary user). The forwarding allows the Directory to hide the responsible AP to the device until it is authenticated.

3. **Authentication Request** The AP now handles the forwarded request based on its internal user table (or creates a temporary user identifier for the user in case of a temporary token request).

4. **Authentication Response** The AP replies with either a bearer token [5] or a failure message which is then forwarded through the Directory to the device. Note that the reply in the case of success is such that the device can identify the AP, whereas in the case of failure, the reply is such that the device cannot identify

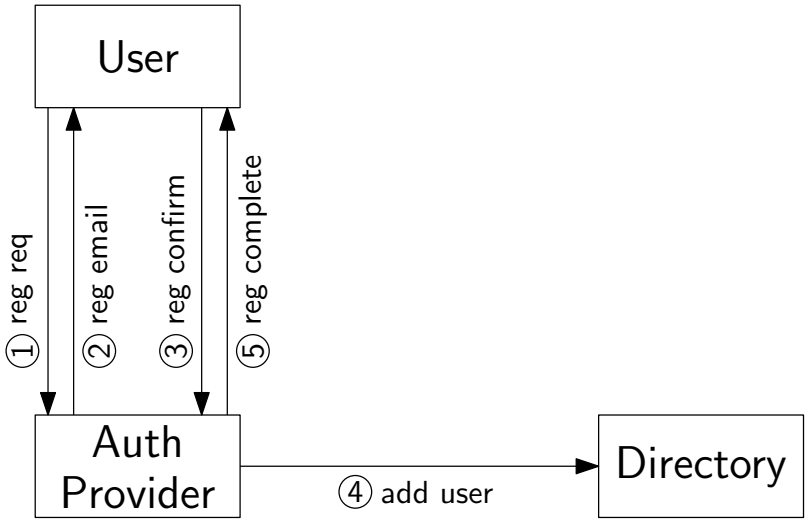


Figure 2: Registration Flow for MediaAUTH

the AP. This prevents malicious users to scan for existing accounts (and their authoritative APs).

The simplified login flow follows the same principle, but no routing is done through the Directory, as the device knows the responsible AP and hence can contact it directly. Note that for temporary tokens, the device may use any AP (but may also use the routed flow, if no AP is known).

3.2 User Registration

A user would like to create a user account on an AP in order to use the fully fledged MediaAUTH service (i.e. paired services). The registration flow is represented in figure 2. In the following, each step is explained.

1. **Registration Request** The user enters his e-mail address and the desired password into his device. The device requests registration from an AP using this information.

2. **Registration E-Mail** The AP sends a confirmation e-mail to the supplied address in order to verify the ownership of the address.

3. **Registration Confirmation** The user clicks on a link in the e-mail in order to certify that he is the owner of the address.

4. **Add User** The AP, having now verified the identity of the user, registers this user with the Directory, mapping this user name to this particular AP. If the user is already registered with another AP, this request fails and the Directory replies with the Fully Qualified Domain Name (FQDN) of the responsible AP.

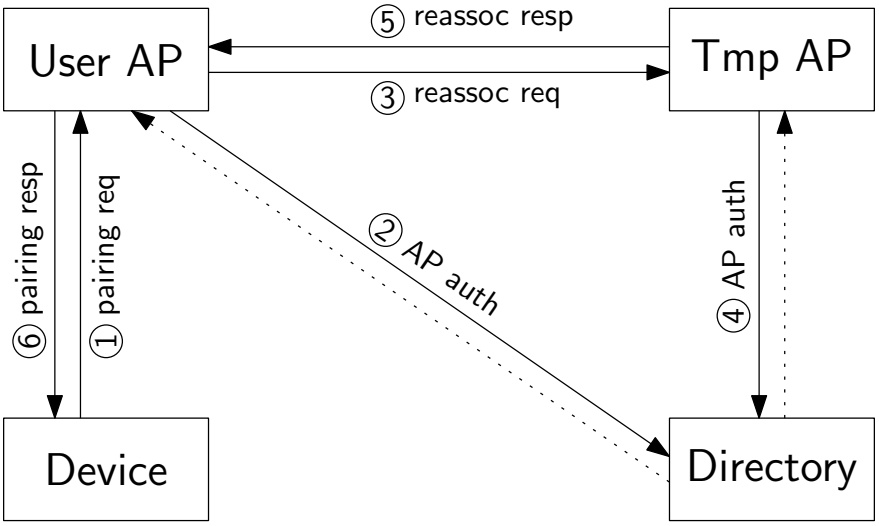


Figure 3: Pairing Flow for MediaAUTH

5. **Registration Complete** Based on the reply of the Directory, the AP gives an informative reply to the user.

3.3 User Pairing

Once a user on device has associated with his authoritative AP and the device holds a temporary user token, the associated temporary identifier should be re-associated to the user in order to grant him access to the resources issued to the temporary identifier (e.g. unpaired tags).

1. **Pairing Request** The device just finished association with the AP but also holds a temporary token. It sends an (authenticated) pairing request to its AP, passing on the temporary token.

2. **AP Authentication** The user's AP first checks with the Directory if the temporary AP is a legitimate AP.

3. **Re-association Request** The user's AP then forwards the temporary token to the temporary AP in a re-association request, asking it to remove the token from its list and return the associated temporary user id.

4. **AP Authentication** The temporary AP now checks the legitimacy of the user's AP with the Directory.

5. **Re-association Response** If the received temporary token is valid, the temporary AP removes it from its database and replies with the temporary user id to the user's AP.

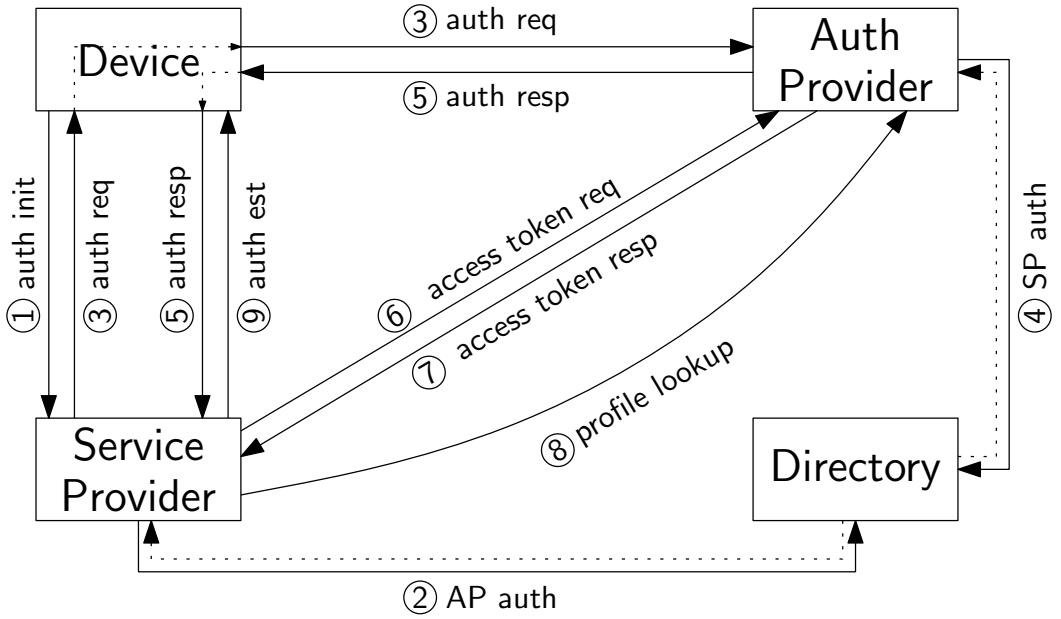


Figure 4: SP Login Flow for MediaAUTH

6. **Pairing Response** The user's AP now adds the received temporary identifier to the user's list of associated temporary identifiers and replies to the device that the pairing was successful.

It is important to realize, that the associations to *all* temporary identifiers have to be stored at the user's AP in order to allow SPs to identify data they have stored using any of the temporary identifiers as this particular user's data. Retaining state about which SP has to be updated about which temporary identifier's re-association seems to be unfeasible.

3.4 SP Login

An authenticated user (registered or temporary) would like to use a particular service of a SP for the first time. Trust between the user/device and the SP has to be established. The authentication flow is represented in figure 4. In the following, each step's meaning and implication is explained.

1. **Authorization Initialization** The device requests the SP to initialize the login procedure by passing on the responsible AP.

2. **AP Authentication** The SP verifies the legitimacy of the AP supplied by the device with the Directory. Note that there exists a trust relationship between the Directory and the SP. Hence the SP can trust the information from the Directory and the Directory can trust that the SP does not abuse the information.

3. **Authorization Request** The SP then issues an authorization request to the AP which is sent through the device. The device must authenticate to the responsible AP and then pass on the request. Note that after authentication of the device at the AP, there exists a trust relationship between the device and the AP.

4. **SP Authentication** The AP now checks if the SP is indeed an authorized SP by checking with the Directory. The same trust relationship as for AP Discovery exists between the AP and the Directory. Note that this does not (yet) authenticate the SP to the AP, since at this point, the SP's identity is only a self-declaration.

5. **Authorization Response** If the user has authenticated successfully and the SP has been verified, the AP sends a code to the SP (through the device), which allows the SP to request a token from the AP.

6. **Access Token Request** The SP sends this code directly (so *not* through the device) to the AP for verification. Using internal state, the AP checks if the token request really comes from the (formerly self-declared) SP by means of Secure Socket Layer (SSL) certificates. If this is the case, it issues a token to the SP. A bi-directional trust relation between the SP and AP exists now.

7. **Access Token Response** If the AP (to which a trust relation exists) replies to the SP with a token, it means that the received code really came from the AP and hence the user/device has been authenticated by the AP. The SP hence may now trust the device.

8. **Profile Lookup** The SP can now – using the token – access relevant profile information of the user (n.b. the user identifier and all associated temporary identifiers). It uses this information to uniquely identify the user and map the users identity to its local database entries.

9. **Authorization Establish** The SP now replies to the device with a bearer token [5], allowing the device to authenticate itself to this SP for a limited time-span. This token can be used in other services such as RadioTAG.

4 Specification

Note: This section will probably still undergo some changes, until the general mechanisms are approved.

The authentication mechanism uses Authorization Code Grant as specified in the OAuth 2.0 specification draft [4, section 4.1] with confidential clients (section 2.1).

4.1 RadioDNS application discovery

To discover MediaAUTH services, the service lookup procedure as described in the RadioDNS standard [13] is used. The SRV record for MediaAUTH is `_mediaauth`.

4.2 Encryption & Certification

At several points, Transport Layer Security (TLS) with client-side SSL certificates is required. Notably in the communication between the AP, the SP and the Directory. Note that in this setup, the *authentication* of these three entities is handled by the SSL certificates, i.e. by an external, trusted, Certificate Authority (CA) just like for example an e-banking site. The certificate SHOULD be issued by a commonly accepted CA (see common browser's and operating system's CA policies [1, 6, 7, 12]). On the other hand, the Directory *authorizes* the AP and SP to use the system by means of their FQDN.

All HTTP requests from the device SHOULD use TLS if the device supports it. Note that the AP and the SP MUST support the use of TLS when communicating with the device. Further, if TLS is used, any server MUST use a valid, signed SSL server-certificate.

4.3 SP Login

Figure 4 represents the authentication flow for MediaAUTH. In the following, every step is described individually. Note that steps (1), (2), (4) are not part of the OAuth 2.0 specification [4]. Step (8) is part of the OAuth 2.0 Bearer-Token specification [5].

With respect to the OAuth 2.0 specification [4], the AP assumes the role of the authorization server, the SP the role of the client and the device the role of the user-agent with attached resource owner.

1. **Authorization Initialization** The device requests via HTTP GET the `http://<host>:<port>/auth?ap=<ap>` resource, where `host` and `port` are obtained by RadioDNS lookup. `ap` is the FQDN of the responsible AP.

2. **AP Discovery** The SP verifies the legitimacy of the AP with the Directory. Details are to be specified.

3. **Authorization Request** Conforming to [4, section 4.1.1.]. The SP MUST redirect the device to `http://<host>/oauth`, where `host` is the AP's location as supplied by the device in step (1), using HTTP status code 302. The SP MUST NOT use the `redirect_uri` field. Upon redirection, the device MUST use the bearer token, obtained with AP login, to authenticate.

4. **SP Authentication** The AP now requests the location of the SP using the `client_id` field of the authorization request. This ensures that the AP does not issue a code to non-authorized service providers. The AP requests via HTTP GET `https://<directory>/verify?client_id=<client_id>`. TLS with client-side certificate MUST be used and the Directory MUST verify the identity of the AP and reply with HTTP status code 200 and the location (FQDN and optional port number) of the SP in the `text/plain` format if it is found, HTTP status code 403 if AP authentication failed and 404 if the SP is not found. In any but the first case, the AP MUST abort the authorization. Note that the AP SHOULD respect

HTTP caching headers and SHOULD use the cached data where possible in order to offload the Directory.

5. **Authorization Response** Conforming to [4, section 4.1.2.]. The AP MUST redirect the device to `http://<host>/code`, where `host` is the location of the SP obtained in step (4).

6. **Access Token Request** Conforming to [4, section 4.1.3.]. The token endpoint is `https://<host>/token`, where `host` is the FQDN (plus optional port number) of the AP as supplied by the Directory in step (2). The SP MUST use TLS with client-side certificate and MUST include the `client_id` parameter (see [4, section 3.2.1.]). The AP MUST NOT issue a client password to the SP and is hence not required to support HTTP basic authentication. The SP is identified using the client-side certificate. Further, the AP MUST verify that the requested code has been issued to the same FQDN as in the SP's certificate.

7. **Access Token Response** Conforming to [4, section 4.1.4.]. The AP MUST respond with a bearer token [5].

8. **Profile Lookup** Conforming to [5, section 2.]. The SP now requests the `https://<host>/profile` resource using its token. The request MUST be issued using TLS with client-side certificate and the AP MUST verify the SP's certificate and that the token has been issued to this particular SP. If authentication succeeds, the AP MUST reply with HTTP status code 200 and the following parameters in the HTTP entity body using the `application/json` media type [2]. All parameters are REQUIRED.

   `user_id` The e-mail address of the user, which uniquely identifies this user. Maybe it is better to use Globally Unique Identifiers (GUIDs) here.

   `tmp_ids` Array of temporary identifiers associated with this user.

9. **Authorization Establish** The SP now replies to the authorization response issued in (5). If it the access token request in step (6) succeeded, it MUST reply with HTTP status code 200 and the following parameters in the HTTP entity body using the `application/json` media type [2]. All parameters are REQUIRED.

   `token` A device token generated by the SP to authenticate and authorize the device to use its services.

   `expires_in` The lifetime in seconds of the access token.

   The SP MUST include the HTTP `Cache-Control` response header field [9] with a value of `no-store` in any response containing tokens, credentials, or other sensitive information, as well as the `Pragma` response header field [9] with a value of `no-cache`.

   If any of the steps after step (5) failed, the SP MUST reply with HTTP status code 403.

If this last step succeeds, the device can use the supplied device token to authenticate to services provided by the SP.

5 Integration with RadioTAG

Integration with RadioTAG [11] can be achieved by omitting the authentication mechanism proposed by RadioTAG and using the bearer token issued by the SP, both with registered users (i.e. paired scope) and temporary users (i.e. unpaired scope). SPs SHOULD allow both authentication methods to be used. Devices SHOULD prefer MediaAUTH if supported as it provides a better user experience.

6 Considerations

6.1 User AP exposure

This issue has been solved by relaying authentication requests through the Directory.

6.2 Unique Device Identification

For some services, it might make more sense to uniquely identify a device, rather than a user (e.g. device configuration). It might be desirable to extend MediaAUTH with capabilities of unique device identification.

References

[1] Apple Inc. Apple root certificate program. `http://www.apple.com/certificateauthority/ca_program.html`.

[2] D. Crockford (JSON.org). Rfc4627: The application/json media type for javascript object notation (json), 2006. `http://tools.ietf.org/html/rfc4627`.

[3] Frontier Silicon. Frontier silicon radio portal. `http://www.wifiradio-frontier.com/`.

[4] D. Hardt. The oauth 2.0 authorization framework, 2012. `http://tools.ietf.org/html/draft-ietf-oauth-v2-31`.

[5] M. Jones (Microsoft), D. Hardt, and D. Recordon (Facebook). The oauth 2.0 authorization framework: Bearer token usage, 2012. `http://tools.ietf.org/html/draft-ietf-oauth-v2-bearer-22`.

[6] Microsoft Corp. Microsoft root certificate program, 2009. `http://technet.microsoft.com/en-us/library/cc751157.aspx`.

[7] Opera Software. Getting your root certificate included in opera, 2012. `http://www.opera.com/docs/ca/`.

[8] Pure. The lounge.
http://www.thelounge.com/.

[9] R. Fielding (UC Irvine), J. Gettys (Compaq/W3C), J. Mogul (Compaq), H. Frystyk (W3C/MIT), L. Masinter (Xerox), P. Leach (Microsoft), and T. Berners-Lee (W3C/MIT). Rfc2616: Hypertext transfer protocol – http/1.1, 1999.
http://tools.ietf.org/html/rfc2616.

[10] S. Bradner (Harvard University). Rfc2119: Key words for use in rfcs to indicate requirement levels, 1997.
http://www.ietf.org/rfc/rfc2119.txt.

[11] S. O'Halpin (BBC R&D) and C. Lowis (BBC R&D). Radiotag specification version 1.00 (draft 1), 2009.
http://radiotag.prototyping.bbc.co.uk/docs/radiotag-api-proposal-v1.00.html.

[12] The Mozilla Project. Mozilla ca certificate policy, 2012.
http://www.mozilla.org/projects/security/certs/policy/.

[13] The RadioDNS Project. Radiodns technical specification, 2012.
http://radiodns.org/documentation/rdns01-1-0-0/.

Glossary

AP Authentication Provider.

CA Certificate Authority.

FQDN Fully Qualified Domain Name.

GUID Globally Unique Identifier.

SP Service Provider.

SSL Secure Socket Layer.

TLS Transport Layer Security.

Contact

European Broadcasting Union – Technical Department