API Authentication explained

# How authentication in Laminas API Tools works (+ our tweaks)

## Problem

Need to make API authentication and normal authentication work together.

## Solution (might be improved)

During routing (more info on the flow below) check if there is an active session in persistent storage. If there is, consider authentication complete. If there isn’t, try to authentication use via HTTP Basic protocol.

## Flow

1. Bootstrap
   1. AuthenticationServiceFabric: “authentication” service created by a specified fabric  
      TWEAK: Officio API2 – we have replaced it with our ApiAuthenticationService to be able to use persistent/non-persistent storage.
   2. Module: MVCAuthEvent initialized
   3. Module: MVCRouteListener initialized and added listeners to ROUTE event:
      1. Authentication
         1. Attached DefaultAuthenticationListener  
            Tweak: Officio API2 – using delegator factory we have replaced Http authentication adapter with our own (details on how it works are in “Routing” section).  
            Note: this listener won’t trigger it’s own authentication in case of regular non-api calls. In order for this authentication work, it has to be configured in the config (mapping controller to authentication type).
      2. Post Authentication
         1. Attached DefaultPostAuthentication Listener
         2. Attached module’s own AuthenticationPost listener
      3. Authorization
         1. Attached DefaultResourceResolverListener
         2. Attached DefaultAuthorizationListener
      4. Post Authorization
         1. Attached DefaultPostAuthorizationListener
2. Routing
   1. Actual routing – figuring out which part of application will handle the request
   2. API middleware:
      1. Http Method override
      2. Authentication
         1. DefaultAuthenticationListener triggered:
         2. Finds proper adapter (HttpAdapter in our case)  
            Tweak: we have replaced HttpAdapter with our own which doesn’t challenge the client and allow empty password, handling username as an access token. Also it’s configured to have custom basic resolver, which is our own Access Token resolver.
         3. Calls AuthenticationService->authenticate() by passing itself as an argument, receives identity, returns it
         4. Set identity to the storage
      3. Post Authentication
         1. DefaultAuthenticationPostListener triggered – sets response 401 and error message in case of authentication failure
      4. Authorization
         1. DefaultResourceListener triggered – determines controller-action or resource-verb responsible for handling the route
         2. DefaultAuthorizationListener triggered – checks if resource-verb pair is authorized in config
      5. Post Authorization
         1. DefaultAuthorizationPostListener triggered – sets response status code and error message if any
      6. Content Type listener – responsible for forming response headers