Offline Authentication

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.0/authentication-security/offline-authentication.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

This tutorial explains a sample implementation of offline user authentication.
The goal of offline authentication is to make the login/logout end-user experience consistent whether the end user works online or offline.

Prerequisites

It is assumed that the developer is familiar with IBM MobileFirst Platform Foundation application development, adapter-based authentication, and JSONStore.
The associated sample project demonstrates an end-to-end scenario. However, there are many ways to achieve the same results.

This tutorial covers the following topics:

- Offline authentication concept
- Implementing the authentication flows
- Sample application

Offline authentication concept

By using offline authentication, an end user is able to authenticate even if no Internet connection is available.

Online login is necessary when logging-in is attempted for the first time, so that MobileFirst Server validates the user's credentials. After successful validation, a JSONStore object is created for the logged-in user. You implement offline authentication by opening the JSONStore object for the supplied username and password. This is further explained below.

The sample application for this tutorial has the following structure:

- An unsecured area, which is accessible without authentication
- A secured area, which is accessible only after authentication, whether online or offline
- The same login form for both online and offline authentication, so that the end-user experience is the same in either case
- Support for logging out

Going through the online and offline flows from a high-level point of view:

Step 1:

Before either online or offline authentication occurs, the availability of an Internet connection is checked first. Depending on the result, the appropriate authentication flow is used.

Online authentication

Steps 2 and 3

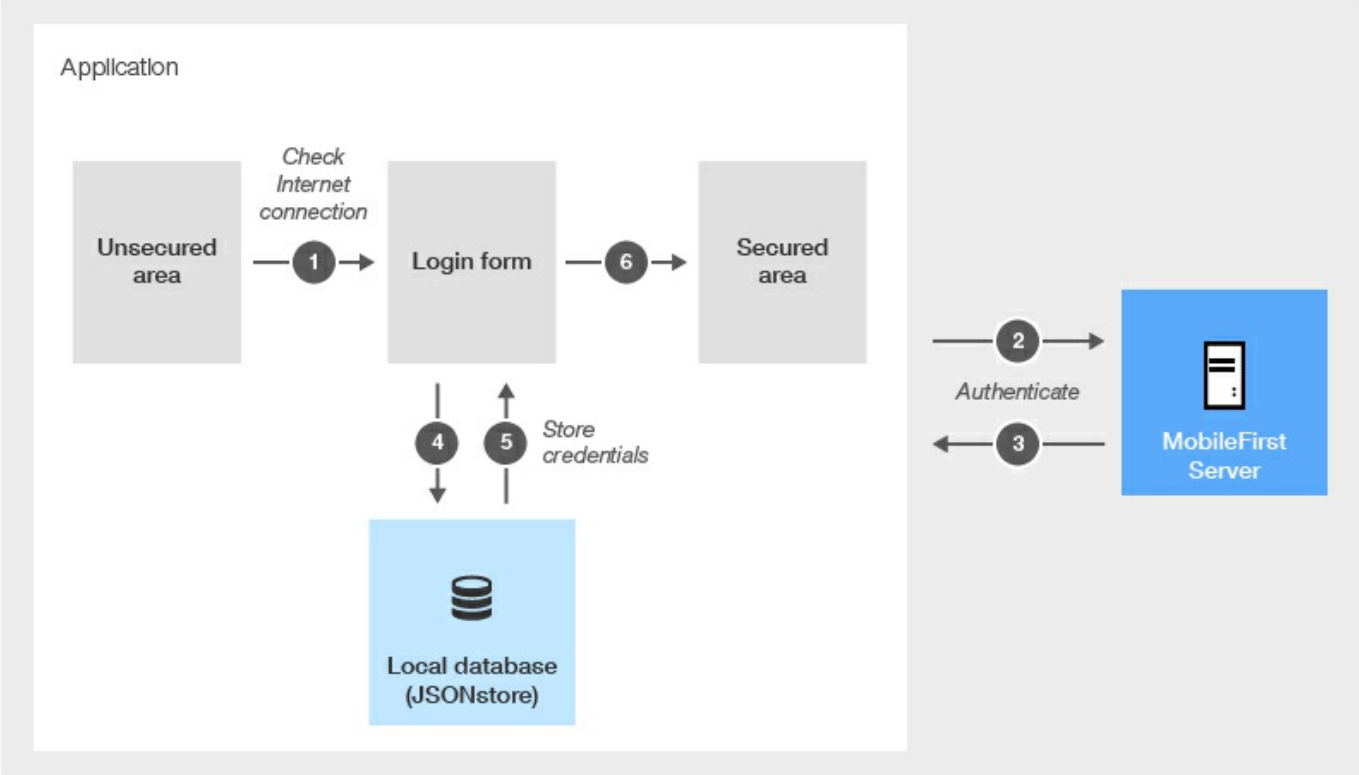
The online authentication flow is implemented as explained in the adapter-based authentication in hybrid applications (../../authentication-security/adapter-based-authentication/adapter-based-authentication-hybrid-applications/) tutorial.

Steps 4 and 5

After successful login, a JSONStore is created by using the end-user's username and encrypted by using the user's password.

Step 6

The secured area is displayed.



Offline authentication

Steps 2 and 3

The authentication process tries to open the previously created JSONStore by using the end-user's password that was passed as input.

Step 4

If the attempt is successful, the secured area is displayed.

Implementing the authentication flows

Note: Some of the implementation code relates to the application UI.
You can find the rest of the application code (HTML, CSS, additional JavaScript) in the sample application.

The `username`, `password` and `offlineLoggedIn` global variables help determine the user's login state, whether connected or not connected to the Internet.

```
var username = null, password = null;
var offlineLoggedIn = false;
```

Prerequisite: When an end user tries to log in for the very first time, an Internet connection must be available so that the user credentials can be verified against the server back end.

Therefore, before any authentication is performed, the Internet connection is checked so that the appropriate authentication flow is executed.

```
function goToSecuredArea() {
    WL.Device.getNetworkInfo(
        function (networkInfo) {
            if (networkInfo.isNetworkConnected == "true")
{
                onlineAuthentication();
            } else {
                offlineAuthentication();
            }
        }
    );
}
```

Implementation notes

Single-Step authentication

1. The online authentication flow uses the same code as the one provided in the single-step adapter authentication sample application (../../authentication-security/adapter-based-authentication/).
2. To simulate a back end of multiple users, a login is considered valid when `username == password`, no matter what `username` is.
   The `submitAuthentication` function in the adapter `*-impl.js` file is therefore modified to:

   ```
   function submitAuthentication (username, password){
       if (username == password) {
   ```

JSONStore

Because the authentication process simulates multiple users, you must use multiple JSONStores, too, in order to accommodate the multiple users.

Thus, in the `WL.JSONStore.init` file, the `username` string is passed as a parameter for a new JSONStore to be opened for each authenticated user. For this, declare a collection to define the structure of the JSONStore:

```
var collectionName = "userCredentials";
var collections = {
    userCredentials : {
        searchFields : {
            collectionNotEmpty: "string"
        }
    }
};
```

Online authentication

Below is the complete implementation, followed by a breakdown and explanation, code block by code block. The sample application contains additional comments.

```
function onlineAuthentication() {
  if (WL.Client.isUserAuthenticated("myCustomRealm")) {
    if (username == null) {
      WL.Client.logout("myCustomRealm", {
        onSuccess: function() {
          onlineAuthentication();
        },
        onFailure: function() {
          WL.SimpleDialog.show("Logout", "Unable to logout at this time. Try again later.", [{
            text: "OK",
            handler: function() {}
          }]);
        }
      });
    } else {
      displaySecuredArea();
    }
  } else if (username) {
    var invocationData = {
      adapter: "authenticationAdapter",
      procedure: "submitAuthentication",
      parameters: [$("#username").val(), $("#password").val()]
    };
    myCustomRealmChallengeHandler.submitAdapterAuthentication(
      invocationData, {
        onSuccess: onlineAuthenticationSuccess,
        onFailure: onlineAuthenticationFailure
      });
  } else {
    WL.Client.login("myCustomRealm", {
      onSuccess: onlineAuthenticationSuccess,
      onFailure: onlineAuthenticationFailure
    });
  }
}
```

1. Check whether the user is already authenticated.
    - If the check returns `true`, display the secured area.
    - If the check returns `false`, display the login form and start authentication via the challenge handler.

```
function onlineAuthentication() {
  if (WL.Client.isUserAuthenticated("myCustomRealm"))
{
    if (username == null) {
      //...
      });
    } else {
      displaySecuredArea();
    }
  } else if (username) {
    //...
  } else {
    WL.Client.login("myCustomRealm", {
      onSuccess: onlineAuthenticationSuccess,
      onFailure: onlineAuthenticationFailure
    });
  }
}
```

2. If the Internet connection is established and the user is authenticated on the back end but the `username` variable is empty, assume that a logout happened offline.
   Log out the user from the realm before trying to log in again online.

```
if (WL.Client.isUserAuthenticated("myCustomRealm")) {
  if (username == null) {
    WL.Client.logout("myCustomRealm", {
      onSuccess: function() {
        onlineAuthentication();
      },
      onFailure: function() {
        WL.SimpleDialog.show("Logout", "Unable to logout at this time. Try again later.", [{
          text: "OK",
          handler: function() {}
        }]);
      }
    });
  } else {
    displaySecuredArea();
  }
}
```

3. If logging-in occurs offline and an Internet connection is then available, log in to the realm without showing the login form.

```
function onlineAuthentication() {
  if (WL.Client.isUserAuthenticated("myCustomRealm")) {
    //...
  } else if (username) {
    var invocationData = {
      adapter: "authenticationAdapter",
      procedure: "submitAuthentication",
      parameters: [$("#username").val(), $("#password").val()]
    };
    myCustomRealmChallengeHandler.submitAdapterAuthentication
(
      invocationData, {
        onSuccess: onlineAuthenticationSuccess,
        onFailure: onlineAuthenticationFailure
      });
  } else {
    //...
  }
}
```

4. If online authentication fails, for example if MobileFirst Server is unavailable, offline authentication is attempted instead.

```
function onlineAuthenticationFailure() {
    offlineAuthentication();
}
```

Online authentication success

Upon successful online authentication, the secured area can be displayed.

The username and password are saved for future use.
A JSONStore for the user is initialized from the user's `username` string and encrypted by using the user's password.

Before populating the collection, it is best to check whether it is already populated from a previous login attempt. The check returns the contents of the collection.

- If the collection is empty, populate it.
- If the collection is not empty, do not populate it again.

Finally, the username and password variables are emptied, the JSONStore is closed and the secured area is displayed.

```
function onlineAuthenticationSuccess() {
    username = $("#username").val();
    password = $("#password").val();
    WL.JSONStore.init(collections, {password:password, username:username, localKeyGen:true}
)
    .then(function() {
        return WL.JSONStore.get(collectionName).findAll();
    })
    .then(function(findAllResult) {
        if (findAllResult.length == 0) {
            // The JSONStore collection is empty, populate it.
            var data = [{isEmpty:"false"}];
            return WL.JSONStore.get(collectionName).add(data);
        }
    })
    .then(function() {
         password = null;
         username = null;
         WL.JSONStore.closeAll();
         displaySecuredArea();
    })
}
```

Offline authentication

In online authentication, if `username != password`, the authentication process stops. In contrast, during offline authentication in JSONStore, you must deal with a couple of additional scenarios:

- Empty username and password fields
- First-time log-in when offline
- Already authenticated users

Empty username and password fields

Empty username and password fields destroy your JSONStore. As a result, users can no longer log in offline until the next online authentication, where the JSONStore is created again if it does not exist.

First-time log-in

First-time log-in cannot happen because there is no JSONStore yet.

```
function offlineAuthentication() {
  $("#username").val('');
  $("#password").val('');
  $("#authInfo").empty();
  // Don't show the login form  if already offline-logged-in via JSONStore, or previously logged-in online (bu
t have yet logged-out).
  if (offlineLoggedIn === true || username) {
    offlineLoggedIn = true;
    displaySecuredAreaOffline();
```

```
  } else {
    // Prepare the login form.
    $("#unsecuredDiv").hide();
    $("#authDiv").show();
    $("#offlineLoginButton").show();
    $("#onlineLoginButton").hide();
    $("#offlineLoginButton").unbind("click");
    $("#offlineLoginButton").bind("click", function() {
      // Don't allow empty username/password field.
      if (($("#username").val() == '') || ($("#password").val() == '')) {
        $("#authInfo").html("Invalid credentials. Please try again.");
      } else {
        WL.JSONStore.init(collections, {
            password: $("#password").val(),
            username: $("#username").val(),
            localKeyGen: true
          })
          .then(function() {
            /*
             * Need to handle the case where logging in when offline for the first time, in which case logging i
n cannot be done as there wouldn't be a
             * JSONStore available yet.
             */
            WL.JSONStore.get(collectionName).count()
              .then(function(countResult) {
                if (countResult == 0) {
                  WL.JSONStore.destroy($("#username").val());
                  $("#authInfo").html("First time login must be done when Internet connection is available.");
                  $("#username").val("");
                  $("#password").val("");
                } else {
                  // Preserve the username and password, set the offline-logged-in flag and dislay the secured
area.
                  offlineLoggedIn = true;
                  username = $("#username").val();
                  password = $("#password").val();
                  displaySecuredAreaOffline();
                }
              })
          })
          .fail(function() {
            $("#authInfo").html("Invalid credentials. Please try again.");
            $("#username").val("");
            $("#password").val("");
          })
      }
    });
  }
}
```

Already authenticated users

1. This function first checks whether the user is already authenticated.

```
if (offlineLoggedIn === true || username) {
  offlineLoggedIn = true;
  displaySecuredAreaOffline();
}
```

If the user is not authenticated, offline authentication is attempted.

2. Check the validity of the user credentials.
   - If they are not valid, display an error message.
   - If they are valid, try to open and decrypt the user's JSONStore by using the user's password.

```
  $("#offlineLoginButton").bind("click", function() {
    // Don't allow empty username/password field.
    if (($("#username").val() == '') || ($("#password").val() == ''))
{
      $("#authInfo").html("Invalid credentials. Please try again.");
    } else {
      WL.JSONStore.init(collections, {
        password: $("#password").val(),
        username: $("#username").val(),
        localKeyGen: true
       })
```

3. Check the contents of the store.
   - If the store is empty, this user did not previously log in: no stored data in the collection. Destroy the store for this username and provide an appropriate error message.

```
          .then(function() {
            /*
             * Need to handle the case where logging in when offline for the first time, in which
case logging in cannot be done as there wouldn't be a
             * JSONStore available yet.
             */
            WL.JSONStore.get(collectionName).count()
              .then(function(countResult) {
                if (countResult == 0) {
                  WL.JSONStore.destroy($("#username").val());
                  $("#authInfo").html("First time login must be done when Internet connection is a
vailable.");
                  $("#username").val('');
                  $("#password").val('');
```

   - If the JSON store is NOT empty, set the `offlineLoggedIn` flag to `true` and display the secured area.

```
        } else {
          // Preserve the username and password, set the offline-logged-in flag and disla
y the secured area.
          offlineLoggedIn = true;
          username = $("#username").val();
          password = $("#password").val();
          displaySecuredAreaOffline();
        }
```

Logout

```
>
function logout() {
  password = null;
  username = null;
  WL.JSONStore.closeAll();
  offlineLoggedIn = false;
  WL.Device.getNetworkInfo(
    function(networkInfo) {
      if (networkInfo.isNetworkConnected == "true") {
        WL.Client.logout("myCustomRealm", {onSuccess: displayUnsecuredArea, onFailure: displayUnsecu
redArea});
      }
    }
  );
  displayUnsecuredArea();
}
```

1. Upon logout, mark the user as logged out:
    1. Empty the username and password variables.
    2. Close the JSONStore.
    3. Set the `offlineLoggedIn` flag to `false`.

```
  password = null;
  username = null;
  WL.JSONStore.closeAll()
;
  offlineLoggedIn = false;
```

2. Check the Internet connection before logging out.
    - If available, log out the user from the realm and display the unsecured area.

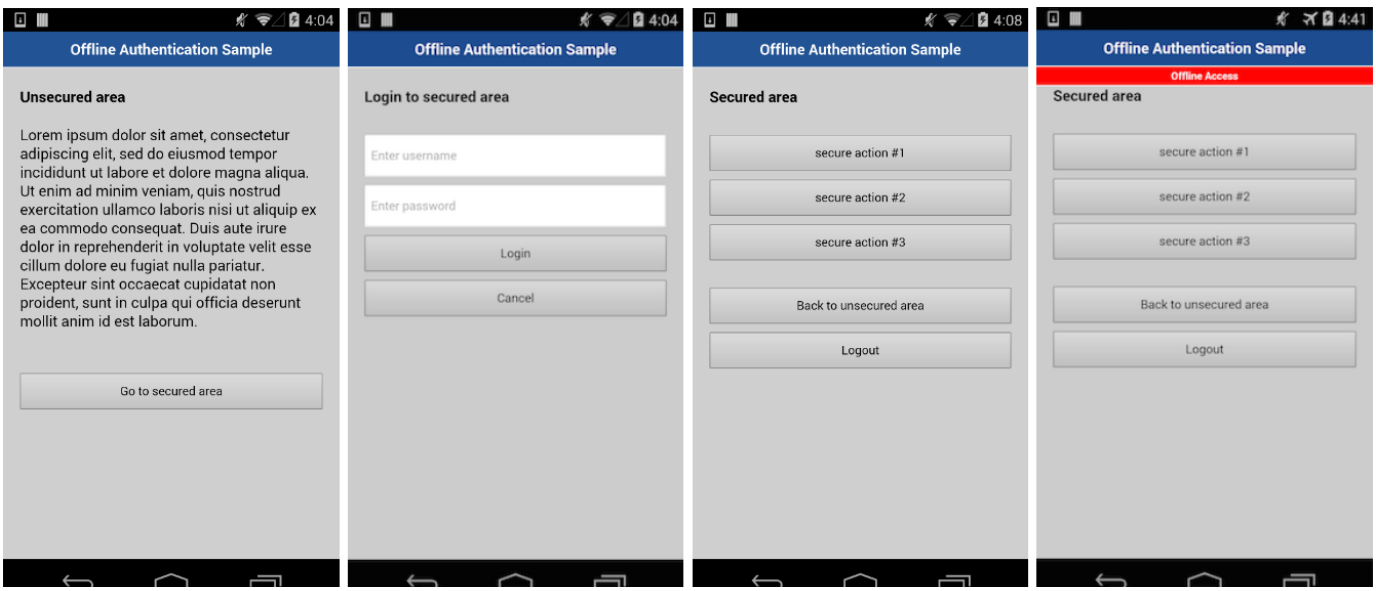
```
WL.Device.getNetworkInfo(
  function(networkInfo) {
    if (networkInfo.isNetworkConnected == "true") {
      WL.Client.logout("myCustomRealm", {onSuccess: displayUnsecuredArea, onFailure:
displayUnsecuredArea});>
    }
  }
);
displayUnsecuredArea();
```

3. If online logout fails (for example because MobileFirst Server is not available), logging out takes place anyway. See online authentication in case the `username` variable is empty.

The end result

Note: The sample code that is written in this tutorial is only a suggestion. You can use other implementation methods.
To use the sample:

1. Import the MobileFirst project into MobileFirst Studio.
2. Deploy the adapter and application.
3. Start the application on a device.
4. Start online and log in by using, for example, `mobile`/`mobile` as the username and password. The secured area is then displayed.
5. Log out.
6. In the device settings, change to airplane mode and log in again. The secured area is then displayed, in "offline access".

Sample application

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/OfflineAuthenticationProject.zip) the Studio project.