NTLM authentication

Overview

The NTLM protocol is a challenge-response mechanism for authentication users in Windows operating systems.
This tutorial explains how to use a MobileFirst adapter when connecting to a back end or resource that is protected by NTLM protocol for user authentication.

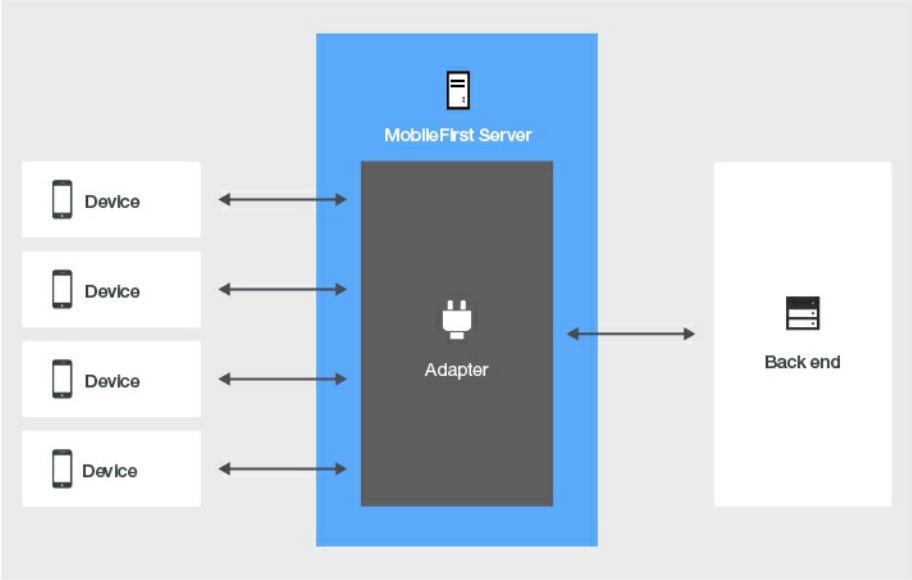
Topics:

- Back-end connection settings (`connectAs="endUser/server"`)
- Using NTLM authentication with `ServerIdentity`
- Using NTLM authentication with `UserIdentity`

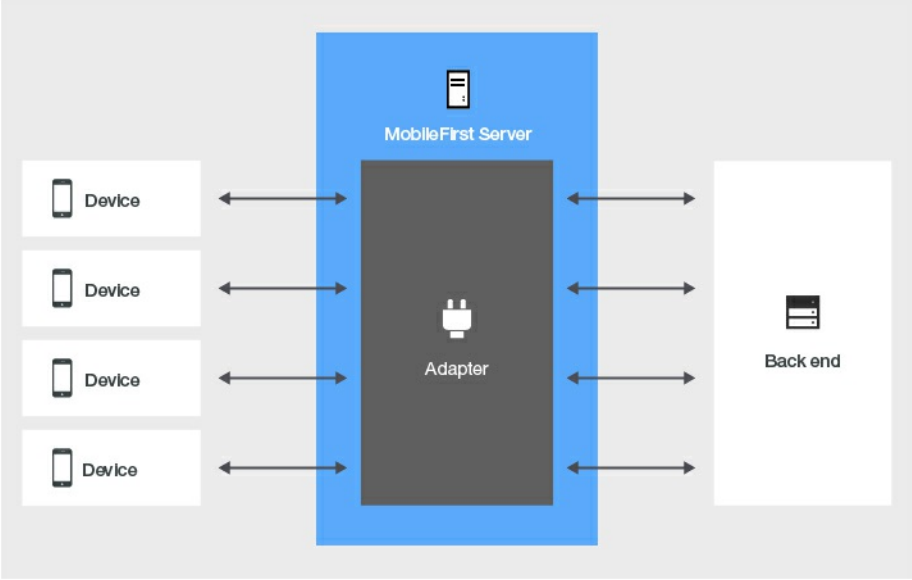
Back-end connection settings (`connectAs="endUser/server"`)

For MobileFirst Server to handle sessions when connecting to a back-end system, you can use either of the following 2 approaches:

- **`connectAS="server"`**:
  All sessions use the same connection context to the back end.
  This is the MobileFirst Server default behavior.



- **`connectAS="endUser"`**:
  Each session is authenticated separately and has a unique connection context against the back end.



For a procedure to be connected as "end user", you must declare it with a `connectAs="endUser"` attribute in the adapter XML file:

```
<procedure name="MyProcedure" connectAs="endUser"/>
```

For more information about the `connectAs` attribute, read the blog post about Configuring HTTP adapters for stateless/stateful back-end connectivity and user identity propagation (https://www.ibm.com/developerworks/community/blogs/worklight/entry/configuring_http_adapters_for_stateless_stateful_backend_connectivity_and_user_identity_p lang=en).

Using NTLM authentication with `ServerIdentity`

When you use a procedure to connect to a back-end server that uses NTLM protocol without specifying the `connectAs` attribute, use the element of the adapter XML file as child element of the element. Also add the child element, so that MobileFirst Server knows which authentication method to use when connecting to the back end.

Make sure to pass the server and user names to the back-end server in the following pattern: *`server-name/user-name`*.

```
<connectivity>
   <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
      <protocol>http</protocol>
      <domain>your-domain-here</domain>
      <port>80</port>
      <connectionTimeoutInMilliseconds>30000</connectionTimeoutInMilliseconds>
      <socketTimeoutInMilliseconds>30000</socketTimeoutInMilliseconds>
      <authentication>
         <ntlm workstation="ServerName"/>
         <serverIdentity>
            <username>your-server-name/your-username-here</username>
            <password>your-password-here</password>
         </serverIdentity>
      </authentication>
      <maxConcurrentConnectionsPerNode>50</maxConcurrentConnectionsPerNode
>
   </connectionPolicy>
</connectivity>
```

Note: When the NTLM protocol is used, the user name is always specified in the *{server-name/user-name}* format.

Using NTLM authentication with `UserIdentity`

Configure MobileFirst Server authentication

1. Create a security test to protect the procedure:

   ```
   <customSecurityTest name="NTLMSecurityTest">
      <test isInternalUserID="true" realm="NTLMAuthRealm"/
   >
   </customSecurityTest>
   ```

2. Use `BasicAuthenticator`, `AdapterBasedAuthenticator`, or any other authenticator that handles `userIdentity`, as the class for the realm used by the security test:

   ```
   <realm name="NTLMAuthRealm" loginModule="AuthLoginModule">
      <className>com.worklight.integration.auth.AdapterAuthenticator</className
   >
      <parameter name="login-function" value="MyAdapter.onAuthRequired"/>
      <parameter name="logout-function" value="MyAdapter.onLogout"/>
   </realm>
   ```

3. Add some login module to create and store user identities to be used by this realm:

   ```
   <loginModule name="AuthLoginModule">
      <className>com.worklight.core.auth.ext.NonValidatingLoginModule</className
   >
   </loginModule>
   ```

4. Add the element and its child element to the adapter XML file, so that MobileFirst Server knows which authentication method to use when connecting to the back end:

```
<connectivity>
   <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
      <protocol>http</protocol>
      <domain>Put.Your.Domain.Here</domain>
      <port>80</port>
      <connectionTimeoutInMilliseconds>30000</connectionTimeoutInMilliseconds>
      <socketTimeoutInMilliseconds>30000</socketTimeoutInMilliseconds>
      <authentication>
         <ntlm workstation="wl-ntlm"/>
      </authentication>
      <maxConcurrentConnectionsPerNode>50</maxConcurrentConnectionsPerNode
>
   </connectionPolicy>
</connectivity>
```

5. Assign this security test to the procedure that is used to connect to the back end protected by NTLM protocol, and add `connectAs="endUser"` to the procedure declaration in the adapter XML file:

```
<procedure name="getNTLMData" securityTest="NTLMSecurityTest" connectAs="endUser"/>
```

Adapter JavaScript code

Create a `UserIdentity` that contains a user identifier and credentials properties. Format the `userId` as *`servername/username`*:

```
function submitAuthentication(username, password){
  var userIdentity = {
    userId: "MyServerName\"+ username,
    credentials: password
  };
  WL.Server.setActiveUser("NTLMAuthRealm", null);
  WL.Server.setActiveUser("NTLMAuthRealm", userIdentity)
;
   ...
}
```

Create an http request to the NTLM protected back end:

```
function getSecretData(){
  var input = {
    method : 'get',
    returnedContentType : 'html',
    path : "index.html"
  };
  return WL.Server.invokeHttp(input)
;
}
```