OFX Implementation Guide

For OFX 2.2 and OAuth

OFX Consortium

Table of Contents

Version History

Date	Author	Version	Description
June 1, 2016	Ernest Riley, Enterprise Engineering, Inc. Rodelio Pagui , Enterprise Engineering, Inc.	1.0.0	Initial Release
July 6, 2016	Ernest Riley, Enterprise Engineering, Inc.	1.0.1	6-23 Edit memo updates

Scope of Document

This implementation guide illustrates best practices and implementation considerations for OFX 2.2 and for out-of-band, security token-based authentication, specifically, OAuth 2.0.

The implementation guide highlights the following:

- Current issues that financial management products and Financial Institutions (FIs) face in acquiring and provisioning users' financial data over the Internet.
- Use of OAuth with OFX 2.2, and how it solves the issues described above.
- Changes in OFX 2.2 spec to accommodate out-of-band OAuth security token.

The intended audience of this implementation guide is expected to have knowledge of OAuth 2.0 framework and the OFX protocol. For more information on OAuth 2.0 and OFX 2.2., please check the Resources section of this document.

OAuth is an Elegant Solution

Below are some of the issues that financial management products face in obtaining a user's financial data over the Internet. This isn't an exhaustive list, but describes the main issues being addressed by this type of implementation.

1. The need to store FI credentials in third-party systems that expose FIs and financial management systems to liabilities associated with potential breach of those credentials.
2. Management and provisioning of credentials across multiple systems result in poor user experience and higher customer service costs. While OFX credentials can be narrowly scoped, e.g., limited to data downloads only, to mitigate the impact of data breach, they still need to be managed. For example, changes in password have to be manually transferred over to any client system that uses the credentials.

OAuth provides an elegant solution to the above problems by ensuring that the customer only ever needs to share credentials with their FI. Once the FI has validated the user credentials, the FI can enable registered OFX clients to access the users' data.

Issues	Solution(s)
Screen scraping of websites	Provide a OFX 2.2 channel using OAuth token-based authentication, which relieves the stress on websites and allows for screen scraping entities to achieve optimal performance.
Authentication performed after OFX request is received by network	OAuth authentication adds an additional layer of security by proxy authentication.

OFX 2.2 Features

Features	Benefits
Open Financial Exchange, OFX	Low-cost delivery channel
Lower costs and simpler implementation	Financial institutions can choose any OFX Solution Provider that supports the specification or develop their own OFX-compliant server to process transactions internally
Freedom of choice	OFX eliminates connectivity as an obstacle to the growth of online banking and financial services, thus promoting competition on the basis of product, price and service.
Choice of services	Financial institutions can choose from banking, bill payment, investments, and bill presentment.
Choice of technology partners	To simplify the process of connecting to customers, financial institutions can choose a system integrator to develop a server or choose a service bureau to outsource banking, bill payment, and other services.
Choice of hardware and software	Financial institutions can use many combinations of hardware and software to support OFX.
Integration of OAuth Support	Using OFX 2.2, personal financial management software can connect with OAuth provided credentials to a financial institution's OFX server. With OAuth, credentials provided by an OAuth provider, or server, will be used to confirm access to perform financial transactions.

OAuth 2.0 Features

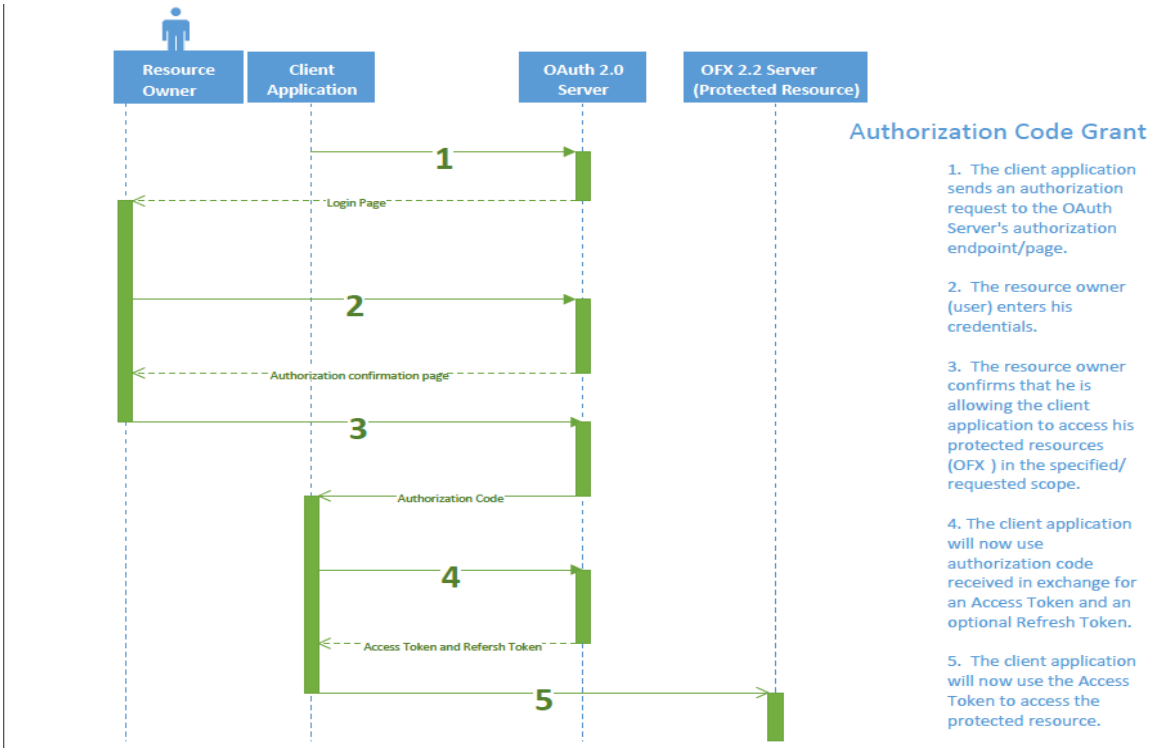
Features	Benefits
Fewer passwords for end-user to remember	Using a well-known OAuth provider, a financial institution gives their clients the option to leverage their existing service credentials to access their financial data.
Reduced Support Surrounding Authentication	Save time and resources surrounding support of authentication.
Reduced Authentication Failures.	Reduced failures related to login will allow support personnel to concentrate efforts elsewhere concerning application support. An addition benefit is that the end-user has one less password to remember.
Permission Management	Easier management of permissions at a client and data level.
Two-legged Authentication	A pre-approved application is allowed to access available resources.
Three-legged Authentication	A resource owner uses a pre-approved application to access available resources.

OAuth 2 Implementation Flow

OAuth 2 Authorization Code Grant

The Authorization Code Grant works in the following manner:

1) The client application sends an authorization request to the OAuth Server's authorization endpoint/page.
2) The resource owner (user) enters his credentials.
3) The resource owner confirms that he is allowing the client application to access his protected resources (OFX) in the specified/requested scope.
4) The client application will now use authorization code received in exchange for an Access Token and an optional Refresh Token.
5) The client application will now use the Access Token to access the protected resource.



Roles

Role	Description
End-User, or Resource Owner	Examples: Financial Institution client accessing FinTech application.
Financial Institution, FI	Brokerages, etc., that hold the end-users' financial data.
Protected Resource, namely OFX 2.2 Server	Interface used by FIs to expose and end-users financial data.
Client Application	Application used by end-user to view their financial data.
OAuth 2.0 Server	Intermediate resource used to prevent the client application having to handle an end-user's credentials, therefore, adding an additional layer of security.

Implementing a Security Token with an OFX 2.2 Server

Part of the two-way exchange for financial data includes authentication. Typically, a username and password is provided as part of each OFX request. As such, the OFX server is responsible for performing the authentication of the provided credentials. With OAuth, the authentication is performed by the OAuth server, leaving the OFX server to perform the financial transaction. One of the features added for OFX 2.2 is the ability to use an OAuth access token in an OFX request.

About ACCESSTOKEN

Servers may require the use of an <ACCESSTOKEN> in place of <USERID> and <USERPASS> for authentication. The use and format of <ACCESSTOKEN> must be arranged out-of-band between the client and the OFX Server provider. Keeping the specific use and format of <ACCESSTOKEN> out-of-band allows OFX to support numerous methods of token generation such as OAUTH 1.0, OAUTH 2.0, JSON Tokens and so on. Essentially any agreed-upon token format and methodology may be used between the client and server.

The intent of <ACCESSTOKEN> is to leverage an out-of-band mechanism that will fully replace all other types of authentication within OFX for all types of accounts and requests. As such, <ACCESSTOKEN> interaction with other <SONRQ> mechanisms and features should be avoided. A client should send ONLY one of the following for authentication:

1. <USERID> and <USERPASS>;
2. <USERKEY>; or
3. <ACCESSTOKEN>

Via out-of-band communication, or outside of the OFX channel, the client application developer acquires the security token from the Financial Institution and is not discussed within this implementation guide. The details of obtaining a security token shall be between the Financial Institution and client application developer.

Signon Realms

As stated in the OFX 2.2 specification, a signon realm:

> *"identifies a set of messages using the same password. Realms are used to disassociate signons from specific services, allowing FIs to require different signons for different messagesets."*

OFX clients may use an OFX Server's profile response to learn of its capabilities. An OFX Server's profile response contains the following sections, which a client can request independently:

- Message Sets – list of services and any general attributes of those services. Message sets are collections of messages that are related functionally. They are generally subsets of what users see as a service.
- Signon realms – FIs can require different signons (please refer to SIGNON section) for different message sets. Because there can only be one signon per <OFX> block, a client needs to know which signon the server requires and then provide the right signon for the right batch of messages.

For OFX clients that use the profile and support out-fo-band security token, the following fields should be present and configured as follows in the SIGNON Realm:

<PINCH>	N since the server will not support <PINCHRQ> (PIN change requests)
<CHGPINFIRST>	Clients must ignore <CHGPINFIRST> since the profile indicates <PINCH>N.
<ACCESSTOKENREQ>	Y if the server requires ACCESSTOKEN for all requests except profile.

The mandatory fields inside the SIGNONINFO aggregate describes the password rules for OFX versions prior to OFX 2.2. For servers that support ACCESSTOKEN instead of user password, these fields can be used to describe the ACCESSTOKEN rules. However, unlike passwords, the client may or may not use these values since the use of ACCESSTOKEN is transparent to the users.

Note that OFX Clients that do not require the profile from the server should consider properly handling unsupported features, specifically, PINCHRQ. For more details, refer to the Important Considerations for Implementations section.

Implementing an OAuth Enabled Client

- All implementations of OAuth clients will result in obtaining a value for an ACCESSTOKEN to be used as part of the OFX request.
- After the security token is obtained via out-of-band communication, and provided to the Financial Institution through other means to obtain an access token, the OFX request shall include SONRQ child element, ACCESSTOKEN, in place of USERID and USERPASS, with its value set to the access token. The OFX server will validate the ACCESSTOKEN as part of the OFX authentication phase. [See Example OFX Sign-On request]
- If ACCESSTOKEN is requested by an OFX server, then status code of 15514 will be returned when only USERID and USERPASS was sent on previous request.
- If an OFX server returns a status code of 15515, then the token authentication failed. OFX client provided a security token that is invalid or unrecognized by the server. This is equivalent OFX client will need to obtain an updated security token via out-of-band communication to use as the ACCESSTOKEN value on the subsequent request.
- If ACCESSTOKEN provided is expired and needs refresh then a 15516 status code is returned. This notifies the OFX client to use whatever out-of-band agreement for handling expired tokens. For instance, an OAuth 2.0 server that supports authorization code grant may require a Refresh Token in exchange of a new security token. While an OAuth 2.0 server that supports implicit grant may require re-authentication to acquire a new security token. [See Brief Overview of OAuth 2 Implementation Framework]

Suggested Practices

- The OAuth implementation should be done after the OFX server is in place and has been certified.
- If possible, the Financial Institution should set-up an OAuth website to allow for application developers to register their application for communication with the OFX server.
- There are two recommended OAuth grants for authentication:
    - Authorization Token; and
    - Implicit Grants
- Credentials grant is highly discouraged since it defeats the purpose of OFX 2.2 security enhancement for anti-password pattern.
- Access Token life span or expiry varies depending on the OAuth server's policy. It is suggested that the process of refreshing an expired token or acquisition of a new token with high consideration on user experience should be discussed between implementers and Financial Institution.
- Even though an OFX server is conceptually a single resource, implementers should still consider establishing specific access token scopes for each OFX message sets. The scopes shall be transparent to the OFX client when sending the access token to the OFX server.

- While it would be permissible under the specification to have multiple signon realms with a mixture of <ACCESSTOKEN> and <USERID>/<USERPASS> used between those realms this is STRONGLY DISCOURAGED due to the client side complexity which would be created.
- While it would be permissible under the specification to use other OFX MFA mechanisms or requests (such as <CHALLENGERQ>) this is STRONGLY DISCOURAGED due to the client side complexity which would be created.

Out-of-Band Communication Topics

- Method of registering an OFX client.
- The number and type of requests needed to obtain token to access resource server, namely the OFX server.
- Define whether a is a token needed per request type, whether a token will act as a "clientid", and token validity duration.
- If necessary, complimentary token transport method (i.e. HTTP Header, POST parameter, etc.).
- Ensuring that end-user credentials aren't visible to API platform.
- And anything else needed for an OFX client to successfully obtain a token.

Step-by-Step Guides for Implementing OAuth with OFX 2.2

Things to do before Enabling OFX 1.x-2.1.x Client and/or Server to Communicate Using ACCESSTOKEN Obtained via OAuth

- Review OFX 2.2 specification
- Contact OFX certification partner
- Assess backwards compatibility
- Financial Institution is expected to have access to an OAuth server

Enable an OFX 1.x-2.1.x Client To Communicate With an OFX 2.2 Server

Goal: To obtain token in order to communicate with resource server, namely an OFX 2.2 server.

1. If applicable, update client to recognize the new OFX Profile response, PROFRS, field is ACCESSTOKENREQ
2. Upon recognizing that the OFX server requires ACCESSTOKEN either from the server, or via Financial Institution out-of-band communication, modify the OFX request by replacing USERID & USERPASS fields with ACCESSTOKEN.
3. Authentication method modifications
    a. Create logic to handle non-zero authentication status codes related to ACCESSTOKEN usage.
        i. Upon receiving status code of 15514, reply with a subsequent request that includes ACCESSTOKEN in place of USERID & USERPASS.
        ii. Upon receiving status code of 15515, obtain the correct ACCESSTOKEN value needed for your client from the FI by predetermined means.
        iii. Upon receiving status code of 15516, obtain an updated ACCESSTOKEN value needed for your client from the FI by predetermined means.
    b. For more information, refer to OFX 2.2 section 2.5.1.7.

Enable an OFX 1.x-2.1.x Server to Communicate With an OAuth Client

Goal: To accept ACCESSTOKEN field as authentication parameter in place of USERID & USERPASS.

1. Update OFX Profile Response (PROFRS)
2. Signon realm changes (see above):
    a. Add ACCESSTOKENREQ tag to SIGNONINFO for each supported messageset.
    b. Specify the value of Y for ACCESSTONKENREQ.
    c. Refer to OFX 2.2 specification section 7.2.2 for more information.
3. Add ability of OFX server to recognize the new field SONRQ field ACCESSTOKEN
    a. If applicable, update, or provide, authentication method to accept and evaluate ACCESSTOKEN value.
    b. Refer to OFX 2.2 specification section 2.5.1.2.3 for more information.
4. Authentication method modifications:
    a. When USERID & USERPASS is specified in the OFX request, return an OFX response with each messagesets STATUS/CODE set to 15514.
    b. When ACCESSTOKEN is specified and is invalid, return an OFX response with each messagesets STATUS/CODE set to 15515.
    c. When ACCESSTOKEN is specified, is valid and requires client to request a new token, return an OFX response with each messagesets STATUS/CODE set to 15516.
    d. For more information, refer to OFX 2.2 section 2.5.1.7.

Sample Implementation

Below are sample OFX server implementations which use out-of-band OAuth 2.0 authentication. Since the implementation guide only illustrates the OFX client and server conversation, the acquisition of access token shall not be covered.

In the sample OFX requests and responses to follow, the OFX XML header is omitted for brevity. For reference, the OFX XML follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<?OFX OFXHEADER="200" VERSION="220" SECURITY="NONE" OLDFILEUID="NONE" NEWFILEUID="NONE"?>
```

Additionally, for formatting purposes, ellipses are used to indicate that the tag value isn't complete and has been shortened.

Sign On

The following sign on implementation illustrates the use of <ACCESSTOKEN> in the <SONRQ>.

Sample Request:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
      <DTCLIENT>20151225</DTCLIENT>
      <ACCESSTOKEN>7c2c362…</ACCESSTOKEN>
      <LANGUAGE>ENG</LANGUAGE>
      <APPID>QWIN</APPID>
      <APPVER>2000</APPVER>
    </SONRQ>
  </SIGNONMSGSRQV1>
</OFX>
```

Sample Response:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
        <MESSAGE>SUCCESS</MESSAGE>
      </STATUS>
      <DTSERVER>20151225164502.967[-5:EST]</DTSERVER>
      <LANGUAGE>ENG</LANGUAGE>
    </SONRS>
  </SIGNONMSGSRSV1>
</OFX>
```

Sign Up (Account Info)

The following OFX client and server conversation illustrates the use of:

1. <ACCESSTOKEN> for sign on;
2. <NAME> to obfuscate account numbers by masking it, or to provide nickname.
3. <ACCTID> to obfuscate account number by replacing it with a reference ID.

Sample Request:

```
<OFX>
  <SIGNONMSGSRQV1>
    --
  </SIGNONMSGSRQV1>
  <SIGNUPMSGSRQV1>
    <ACCTINFOTRNRQ>
      <TRNUID>1525637-36183006-8919-21774</TRNUID>
      <ACCTINFORQ>
        <DTACCTUP>19900101</DTACCTUP>
```

Sample Response:

```
<OFX>
  <SIGNONMSGSRSV1>
  ---
  </SIGNONMSGSRSV1>
  <SIGNUPMSGSRSV1>
   <ACCTINFOTRNRS>
    <TRNUID>1525637-36183006-891</TRNUID>
    <STATUS>
     ---
    </STATUS>
    <ACCTINFORS>
      <DTACCTUP>20150911125700.833[-4:EDT]</DTACCTUP>
      <ACCTINFO>
        <NAME>5475-****-****-8474</NAME>
        <CCACCTINFO>
          <CCACCTFROM>
            <ACCTID>46555</ACCTID>
          </CCACCTFROM>
          <SUPTXDL>Y</SUPTXDL>
          <XFERSRC>N</XFERSRC>
          <XFERDEST>N</XFERDEST>
          <SVCSTATUS>ACTIVE</SVCSTATUS>
        </CCACCTINFO>
      </ACCTINFO>
      <ACCTINFO>
        <NAME>*****1059</NAME>
        <BANKACCTINFO>
          <BANKACCTFROM>
            <BANKID>053112615</BANKID>
            <ACCTID>56468</ACCTID>
```

Credit Card Statement Download

The following OFX client and server conversation shows the use of:

1. Credit card statement download.
2. <ACCESSTOKEN> for sign on; and
3. Account Reference ID for the <ACCTID>.

Sample Request:

```
<OFX>
  <SIGNONMSGSRQV1>
    ---
  </SIGNONMSGSRQV1>
  <CREDITCARDMSGSRQV1>
    <CCSTMTTRNRQ>
      <TRNUID>1525637-36180806-7724-20272</TRNUID>
      <CCSTMTRQ>
        <CCACCTFROM>
          <ACCTID>46555</ACCTID>
        </CCACCTFROM>
        <INCTRAN>
          <INCLUDE>Y</INCLUDE>
        </INCTRAN>
      </CCSTMTRQ>
    </CCSTMTTRNRQ>
  </CREDITCARDMSGSRQV1>
</OFX>
```

Sample Response:

```
<OFX>
  <SIGNONMSGSRSV1>
   ---
  </SIGNONMSGSRSV1>
  <CREDITCARDMSGSRSV1>
    <CCSTMTTRNRS>
      <TRNUID>1525637-36180806-7724-20272</TRNUID>
      <STATUS>
        <CODE>0</CODE>
        <SEVERITY>INFO</SEVERITY>
        <MESSAGE>SUCCESS</MESSAGE>
      </STATUS>
      <CCSTMTRS>
        <CURDEF>USD</CURDEF>
        <CCACCTFROM>
          <ACCTID>13250I00-00008111</ACCTID>
        </CCACCTFROM>
        <BANKTRANLIST>
          <DTSTART>20150602100000.000[-4:EDT]</DTSTART>
          <DTEND>20150629100000.000[-4:EDT]</DTEND>
          <STMTTRN>
            <TRNTYPE>DEBIT</TRNTYPE>
            <DTPOSTED>20150602100000.000[-4:…</DTPOSTED>
            <TRNAMT>-37.62</TRNAMT>
            <FITID>M20150603020910ir2sab-000325</FITID>
            <NAME>PP*AWELDYSS</NAME>
          </STMTTRN>
          <STMTTRN>
            <TRNTYPE>DEBIT</TRNTYPE>
            <DTPOSTED>20150603100000.000[-4:…</DTPOSTED>
```

Bank Statement Download

The following OFX client and server conversation shows the use of:

4. Checking Account statement download.
5. <ACCESSTOKEN> for sign on; and
6. Account Reference ID for the <ACCTID>.

Sample Request:

```
<OFX>
  <SIGNONMSGSRQV1>
    --
  </SIGNONMSGSRQV1>
<BANKMSGSRQV1>
    <STMTTRNRQ>
      <TRNUID>1525637-36180806-11458-25979</TRNUID>
      <STMTRQ>
        <BANKACCTFROM>
          <BANKID>053112615</BANKID>
          <ACCTID>45962</ACCTID>
          <ACCTTYPE>CHECKING</ACCTTYPE>
        </BANKACCTFROM>
        <INCTRAN>
          <INCLUDE>Y</INCLUDE>
        </INCTRAN>
      </STMTRQ>
    </STMTTRNRQ>
  </BANKMSGSRQV1>
</OFX>
```

Sample Response:

```
<OFX>
  <SIGNONMSGSRSV1>
    ---
  </SIGNONMSGSRSV1>
  <BANKMSGSRSV1>
  <STMTTRNRS>
    <TRNUID>1525637-36180806-11458-25979</TRNUID>
    <STATUS>
      <CODE>0</CODE>
      <SEVERITY>INFO</SEVERITY>
      <MESSAGE>SUCCESS</MESSAGE>
    </STATUS>
    <STMTRS>
      <CURDEF>USD</CURDEF>
      <BANKACCTFROM>
        <BANKID>053112615</BANKID>
        <ACCTID>45962</ACCTID>
        <ACCTTYPE>CHECKING</ACCTTYPE>
      </BANKACCTFROM>
      <BANKTRANLIST>
        <DTSTART>20150317210300.000[-4:EDT]</DTSTART>
        <DTEND>20150813210600.000[-4:EDT]</DTEND>
        <STMTTRN>
          <TRNTYPE>CREDIT</TRNTYPE>
          <DTPOSTED>20150428205300.000[-4:...</DTPOSTED>
          <TRNAMT>236371.98</TRNAMT>
          <FITID>215308-000344</FITID>
          <MEMO>RIGNET, INC./PAYMENTJNL</MEMO>
        </STMTTRN>
        <STMTTRN>
          <TRNTYPE>CREDIT</TRNTYPE>
          <DTPOSTED>20150428205300.000[-4:EDT]</DTPOSTED>
          <TRNAMT>22624.26</TRNAMT>
          <FITID>215308-000346</FITID>
          <MEMO>COMPUTER ASSOCIA/TRADE PAYM</MEMO>
        </STMTTRN>
        <STMTTRN>
          <TRNTYPE>CREDIT</TRNTYPE>
          <DTPOSTED>20150428205300.000[-4:EDT]</DTPOSTED>
```

Error Handling

The following new error codes were added for OFX 2.2.0. These error codes were specifically added for ACCESSTOKEN authentication.

Value	*Meaning*	*Condition*
15514	<ACCESSTOKEN> required (ERROR)	OFX Server requires ACCESSTOKEN for authentication (ERROR)
15515	<ACCESSTOKEN> unrecognized (ERROR)	Authentication failed; ACCESSTOKEN provided is invalid/unrecognized by the server (ERROR)
15516	<ACCESSTOKEN> expired (ERROR)	ACCESSTOKEN provided is expired and needs refresh (ERROR) *This notifies the client to use whatever out-of-band agreement for handling expired tokens was agreed upon during the implementation.*

Example <CODE>15514 Response

This error was generated since the request had VERSION="220" in the header and the OFX Server implementation is requiring the authentication field <ACCESSTOKEN>:

Sample Request:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
      <DTCLIENT>20151225</DTCLIENT>
      <USERID>DEMOBK</USERID>
      <USERPASS>Password123</USERPASS>
      <LANGUAGE>ENG</LANGUAGE>
      <APPID>QWIN</APPID>
      <APPVER>2000</APPVER>
    </SONRQ>
  </SIGNONMSGSRQV1>
</OFX>
```

Sample Response:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
      <STATUS>
        <CODE>15514</CODE>
        <SEVERITY>ERROR</SEVERITY>
        <MESSAGE>ACCESSTOKEN required for SIGNON.</MESSAGE>
      </STATUS>
      <DTSERVER>20151225164502.967[-5:EST]</DTSERVER>
      <LANGUAGE>ENG</LANGUAGE>
    </SONRS>
  </SIGNONMSGSRSV1>
</OFX>
```

Example <CODE>15515 Response

The OFX client sent an unrecognized or invalid ACCESSTOKEN. This may be caused by wrong security token scope; a token issued for another client application; or incorrect security token. This error code should not be used for expired tokens.

Sample Request:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
      <DTCLIENT>20151225</DTCLIENT>
      <ACCESSTOKEN>7c2c36…</ACCESSTOKEN>
      <LANGUAGE>ENG</LANGUAGE>
      <APPID>QWIN</APPID>
      <APPVER>2000</APPVER>
    </SONRQ>
  </SIGNONMSGSRQV1>
</OFX>
```

Sample Response:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
      <STATUS>
        <CODE>15515</CODE>
        <SEVERITY>ERROR</SEVERITY>
        <MESSAGE>ACCESSTOKEN unrecognized.</MESSAGE>
      </STATUS>
      <DTSERVER>20151225164502.967[-5:EST]</DTSERVER>
      <LANGUAGE>ENG</LANGUAGE>
    </SONRS>
  </SIGNONMSGSRSV1>
</OFX>
```

Example <CODE>15516 Response

The OFX client sent an expired ACCESSTOKEN. If the OAuth server supports refresh then the 15516 status code should be returned by the OFX server.

Sample Request:

```
<OFX>
  <SIGNONMSGSRQV1>
    <SONRQ>
      <DTCLIENT>20151225</DTCLIENT>
      <ACCESSTOKEN>7c2c36…</ACCESSTOKEN>
      <LANGUAGE>ENG</LANGUAGE>
      <APPID>QWIN</APPID>
      <APPVER>2000</APPVER>
    </SONRQ>
  </SIGNONMSGSRQV1>
</OFX>
```

Sample Response:

```
<OFX>
  <SIGNONMSGSRSV1>
    <SONRS>
      <STATUS>
        <CODE>15516</CODE>
        <SEVERITY>ERROR</SEVERITY>
        <MESSAGE>ACCESSTOKEN has expired. Refresh required.</MESSAGE>
      </STATUS>
      <DTSERVER>20151225164502.967[-5:EST]</DTSERVER>
      <LANGUAGE>ENG</LANGUAGE>
    </SONRS>
  </SIGNONMSGSRSV1>
</OFX>
```

Important Considerations for Implementations

- An incorrect OFX Header can result in a bad request or HTTP 400. Only OFX headers with VERSION="220" are accepted for OFX 2.2. When sending OFX 2.2 requests, the XML compliant OFX header should be:

  <?OFX OFXHEADER="200" VERSION="220" SECURITY="NONE" OLDFILEUID="NONE" NEWFILEUID="NONE" ?>

- One of the main goals of OFX 2.2.0 is enhanced security. Token-based authentication, like OAuth, is moving away from the traditional authentication model of USERID and PASSWORD combination. The following should be implemented by OFX 2.2 servers:
  - When <ACCESSTOKEN> is required by an OFX 2.2 server and indicated within the FI Profile's signon realm, servers should set <PINCH> and <CHGPINFIRST> to N to indicate that OFX pin change functionality is not supported. All other pin characteristics should be set to some default value as they are not used with this authentication method.
  - OFX 2.2 Servers should NEVER use <CODE>15000 to request a client side pin change.
  - OFX 2.2 clients should NEVER, even if a server indicates that pin change is supported, send a <PINCHGRQ> message to a signon realm on a server whose profile indicates that <ACCESSTOKEN> is required.
  - If the OFX request has an OFX 2.2 header but sends a PINCHGRQ then the server should return General Error <CODE>2000 with a <MESSAGE> stating that <PINCH> request is unsupported by the server.
- Since XML is the basis for OFX 2.0 and later, OFX 2.2.0 should be XML compliant. Special characters in OFX 2.2 are handled according to the XML standard. Characters such as less than (<), greater than (>), ampersand (&), single quote (') and double quotes (") are predefined in XML and should be escaped if used within a character string. For example, if AT&T is used as text for the NAME element, then it should be represented as <NAME>AT&T</NAME>. Other character strings with many special characters should be enclosed in a CDATA section.

  Note: The space macro ( ) should be used if leading or trailing blanks are meant to be preserved as part of a data element's value. Alternatively, a CDATA block may be used to force the handling of leading or trailing spaces. No special formatting of space characters in the middle of an element's text value is needed.

- Like previous OFX versions, OFX 2.2 is case sensitive. OFX requires upper case letters for tag names and enumerated values. The wrong case would result in a bad request or HTTP 400.
- If the OAuth server requires specific scopes for the access token, these scopes should be defined properly for the OFX server. An incorrect scope may result in authentication failure even if the access token is valid. For instance, if an access token was issued for a statement download scope but used as an access token for a bill pay request then the request will fail.

Appendix

Acronyms

Acronym	Definition	Reference
OFX	Open Financial Exchange	http://ofx.net
FI	Financial Institution	
JSON	JavaScript Object Notation	http://json.org

Glossary

Term	Definition
OAuth	An authorization framework.
Refresh Token	Approved token used to obtain a new access token. Format varies.
Access Token	Approved token used to access protected resources. Format varies.

Resources

OFX Home http://ofx.net

The OAuth 2.0 Authorization Framework – RFC 6749 https://tools.ietf.org/html/rfc6749