dxFeed Market Data:
Token Based Authorization

September 2019

v 1.1

Information in this document is subject to change without notice and does not represent a commitment on the part of dxFeed.

Please visit www.dxfeed.com for latest information updates.

Contents

1. Introduction

1.1. Purpose

This document is an addition to dxFeed Java/C/C# APIs and describes connection using token based authorization.

1.2. Scope

To provide your customers with dxFeed market data, you can connect to our market data feeds with token based authentication. It ensures that each request to a server is accompanied by a signed token, which the server verifies for authenticity and only then responds to the request. Thus, your clients get access to allowed market data feeds for specified period. We do not get your clients' personal details.

We recommend to generate arf new token each 24 hours.

2. Token

2.1. Token format

Self-signed token format:

```
     token := encoded-payload "." signature
     encoded-payload := BASE64(UTF8(payload))
     signature := BASE64(HMAC-SHA256(encoded-payload,
UTF8(secret)))
     payload := issuer "," subject "," not-before-time ","
     expiration-time "," issued-at-time "," message,
```

where:

- `secret` – the key used for signature validation
- `issuer` – principal that issued this token
- `subject` - subject of this token
- `not-before-time` - time when token get to be valid
- `expiration-time` - expiration time of the token, after which token will not be valid
- `issued-at-time` - when this token was issued
- `message` - any string (or '<any string>, <filter_1>;…;<filter_n>' if specified by dxFeed team)

All time variables are specified in seconds from the epoch, in UTC.

For additional information, please check:

- UTF-8
- Base64
- HMAC-SHA256

2.2. Java implementation

Set classpath to folder with auther-api.jar. You get access to the library:

- `com.devexperts.mdd.auth.entitle.EntitleLoginHandlerFactory` - login handler factory that plugs into dxFeed API via manifest (by implementing `com.devexperts.qd.qtp.auth.QDLoginHandlerFactory` service interface - see META-INF/ folder)
- `com.devexperts.mdd.auth.entitle.SampleClient` - sample client code to test the connection. In order to use login handler factory, connection string must specify [login=entitle] parameter.
- `com.devexperts.mdd.auth.util.SignedToken` - it is used to create tokens signed by shared secret.

To generate the token, please use the following code:

```
return SignedToken.newBuilder()
    .setIssuer(issuer)
    .setSubject(sessionType)
    .setMessage(user)
    .setIssuedNow()
    .setExpirationFromNow(Duration.ofDays(1))
    .toToken()
    .signToken(secret);
```

Where

- `issuer` – principal that issued this token
- `sessionType` – session provided by dxFeed team
- `user` – user ID or '<user ID>,<filter_1>; … ; <filter_n>' if feeds are specified for your connection by dxFeed team (e.g., 'testuser,opra;cme')
- `secret` – the key used for signature validation

2.2.1. Error handling

You can handle errors from the server. For this, please change `EntitleLoginHandlerFactory.AutherLoginHandler#login(String)` to react to errors from dxFeed by analyzing `reason` parameter. To find session duplicate events, search for **Duplicate session** text in the error description.

2.3. C# implementation

For C#, please generate a token with this code:

```
using System;
using System.Security.Cryptography;
using System.Text;

namespace AutherTokenGen {
    internal class Program {
        private const string DefaultIssuer = "<name>";
        private const string DefaultSessionType = "<session>";
        private const string DefaultMessage = "<message>";
        private const string DefaultSecret = "<secret>";
        private const int DefaultDays = 1;

        private static readonly UTF8Encoding Encoding
= new UTF8Encoding();

        private static byte[] GetUtf8Bytes(string s) {
            return new UTF8Encoding().GetBytes(s);

        }

        private static string ToBase64(string s) {
            return ToBase64(Encoding.GetBytes(s));
        }

        private static string ToBase64(byte[] bytes) {
            return Convert
                .ToBase64String(bytes)
                .TrimEnd('=')
                .Replace('+', '-')
                .Replace('/', '_');
        }

        public static void Main(string[] args) {
            const string help = "AutherTokenGen <issuer> <sessionType>
<user|message> <secret> <days>";

            var issuer = DefaultIssuer;
            var sessionType = DefaultSessionType;
            var message = DefaultMessage;
            var secret = DefaultSecret;
            var days = DefaultDays;
```

```
            switch (args.Length) {
                case 4:
                case 5: {
                    issuer = args[0];
                    sessionType = args[1];
                    message = args[2];
                    secret = args[3];

                    if (args.Length == 5) {
                        if (!int.TryParse(args[4], out days)) {
                            days = DefaultDays;
                        }
                    }

                    break;
                }

                case 0:
                    Console.WriteLine($"Usage: \n{help}\n");
                    break;
                default:
                    Console.WriteLine($"Usage: \n{help}\n");

                    return;
            }

            Console.WriteLine(
                $"Issuer = {issuer}, SessionType = {sessionType},
User|Message = {message}, Secret = {secret}, Days = {days}\n");

            var now = DateTime.UtcNow;
            var expiration = now.AddDays(days);
            var nowSeconds
= new DateTimeOffset(now).ToUnixTimeSeconds();
            var expirationSeconds
= new DateTimeOffset(expiration).ToUnixTimeSeconds();
            var payload =
$"{issuer},{sessionType},,{expirationSeconds},{nowSeconds},{message}";

            Console.WriteLine($"Payload: {payload}\n");
            var encodedPayload = ToBase64(payload);
            var hmac = new HMACSHA256(GetUtf8Bytes(secret));
            var signature =
ToBase64(hmac.ComputeHash(GetUtf8Bytes(encodedPayload)));
```

```
            var token = $"{encodedPayload}.{signature}";
            Console.WriteLine($"Token: {token}\n");
        }
    }
}
```

2.4. Token generation sample

To test token generation, you can use auther.jar in command line mode:

```
java -cp auther.jar com.devexperts.mdd.auth.entitle.SignedTokenGenerator
<name> <session> test
uithoophaivahG3aa2uS2eu9eich6aef2JaeTh2rus7Vaec7SeeNgunaexaefini
```

Output:

```
ZnhzdHJlZXQscmVhbHRpbWUsLDE1NTkyMzA5MzMsMTU1OTE0NDUzMyx0ZXN0.DIkBUkhgiNa0Bsm
bgo0vGhp78KIjPGT80PlG3W7f3IY
```

3. Establishing connection

Specify the generated token when establishing a connection. Use the following functions:

3.1. For Java API

String address = "127.0.0.1:7501[login=entitle:" + token + "]";
Or (preferably) set global variable and use shorter connection string
`AutherLoginHandlerFactory.setAppToken(token);`
`address = "127.0.0.1:7501[login=entitle]";`

3.2. For C API

`dxf_create_connection_auth_bearer()` function

3.3. For C# API

`NativeConnection(string address, string token, Action<IDxConnection> disconnectListener)` constructor of `NativeConnection` class

3.4. For web service

- For JavaScript (dxfeed.cometd.js library):

```
dx.feed.setAuthToken(token);
```

- For Websocket (on ws-handshake):

```
{ ext: { "com.devexperts.auth.AuthToken": <token> }}
```

- For REST:
  - Specify token in the HTTP-header:
    `Authorization: Bearer <token>`
  - (Deprecated workaround) If a header cannot be set, use URL-parameter:
    `<REST-method>.json?access_token=<token>`

4. Testing

To test token generation and data request, please use this command (for Java):

```
java -Dentitle=<issuer>,<session-name>,<user-id>
-DentitleSecret=<session-secret>
-jar auther-api.jar <host:port>[login=entitle] Quote IBM,GOOG,AAPL
```

Where:

- `issuer` – principal that issued this token
- `session-name` — session provided by dxFeed team
- `user-id` — end-user identification
- `secret` – the key used for signature validation
- `-jar auther-api.jar` — path to auther-api (auther-api/lib/auther-api.jar)
- `<host:port>[login=entitle]` — endpoint provided by dxFeed support team

Example:

```
java -Dentitle=acme,demo,1234 -DentitleSecret=0123456789
-jar auther-api.jar localhost:7501[login=entitle] Quote IBM,GOOG,AAPL
```

5. Revision history

Revision	Date	Changes
1.0	April 5, 2019	Initial version
1.1	September 5, 2019	• Added code for token generation for C# (section 2.3) • Extended description of connection establishing for web services (section 3.4) • Added description for fields for connection testing (section 4) • Added revision history