Contents

Securing your OData service comes in many forms. Here we will show you how to secure it using basic authentication and authorization headers. You can read about other methods to secure your OData service here.

Note: This topic assumes that you understand how to enable OData.

- **Platform:** N/A
- **Language:** C#
- **Download:** Basic Authentication with ODataTour

Problem

You can secure your OData service with basic authentication using a custom DevForce *IEntityLoginManager*.

Solution

When you want to query your DevForce entities through an OData service, you usually want to create a *DataServiceContext* by passing in the service Uri as follows:

```
Uri serviceRootUri = new Uri("http://localhost:9009/ODataService.svc");
var context = new NorthwindIBEntities(serviceRootUri);
var customer = context.Customers.FirstOrDefault();
```

The *DataServiceContext* has a *SendingRequest* event that is triggered when an Http request has been created. In this case, the Http request is triggered on our Customers query. The *SendingRequest* event is where we want to create and add our authorization headers.

```
context.SendingRequest += (o, requestEventArgs) => {
  var creds = username + ":" + password;
  var encodedCreds = Convert.ToBase64String(Encoding.ASCII.GetBytes(creds));
  requestEventArgs.RequestHeaders.Add("Authentication", "Basic" + encodedCreds);
};
```

Notice that we're using basic authentication by encoding our credentials with base-64 digits.

On the server, we can intercept and validate the header in our LoginManager as follows:

```
public class MyLoginManager : IEntityLoginManager {
  public IPrincipal Login(ILoginCredential credential, EntityManager entityManager) {
    // Get the authentication header
    string authHeader = HttpContext.Current.Request.Headers["Authentication"];

    //Parse authentication header here
    var creds = ParseAuthHeader(authHeader);
    if (creds == null) {
      throw new LoginException(LoginExceptionType.NoCredentials, "Please supply login credentials");
     }
    // Validate the supplied credentials.
    var aNewCredential = new LoginCredential(creds[0], creds[1], "");
    if (!IsCredentialValid(aNewCredential ))
      throw new LoginException(LoginExceptionType.Other, "Invalid username or password");
    return new UserBase(new UserIdentity(aNewCredential .UserName));
   }
}
```

Note that the *ILoginCredential* supplied to the method will be null. We created a second credential here to make passing the credentials to our validation routine easier, but all that DevForce requires is that you return an *IPrincipal*, such as the *UserBase*, from the Login method.

There's one other change you'll need to make to the *DataService*. The *EntityManager* will by default use the singleton *DefaultAuthenticationContext*, so to ensure that the credentials for each request are correctly validated we need to disable this feature. We do so by overriding the *CreateDataSource* method, as shown below:

```
protected override NorthwindIBEntities CreateDataSource() {
    var em = base.CreateDataSource();
```

```
    em.Options.UseDefaultAuthenticationContext = false;
    return em;
    }
```