Security With OAuth 1.0a

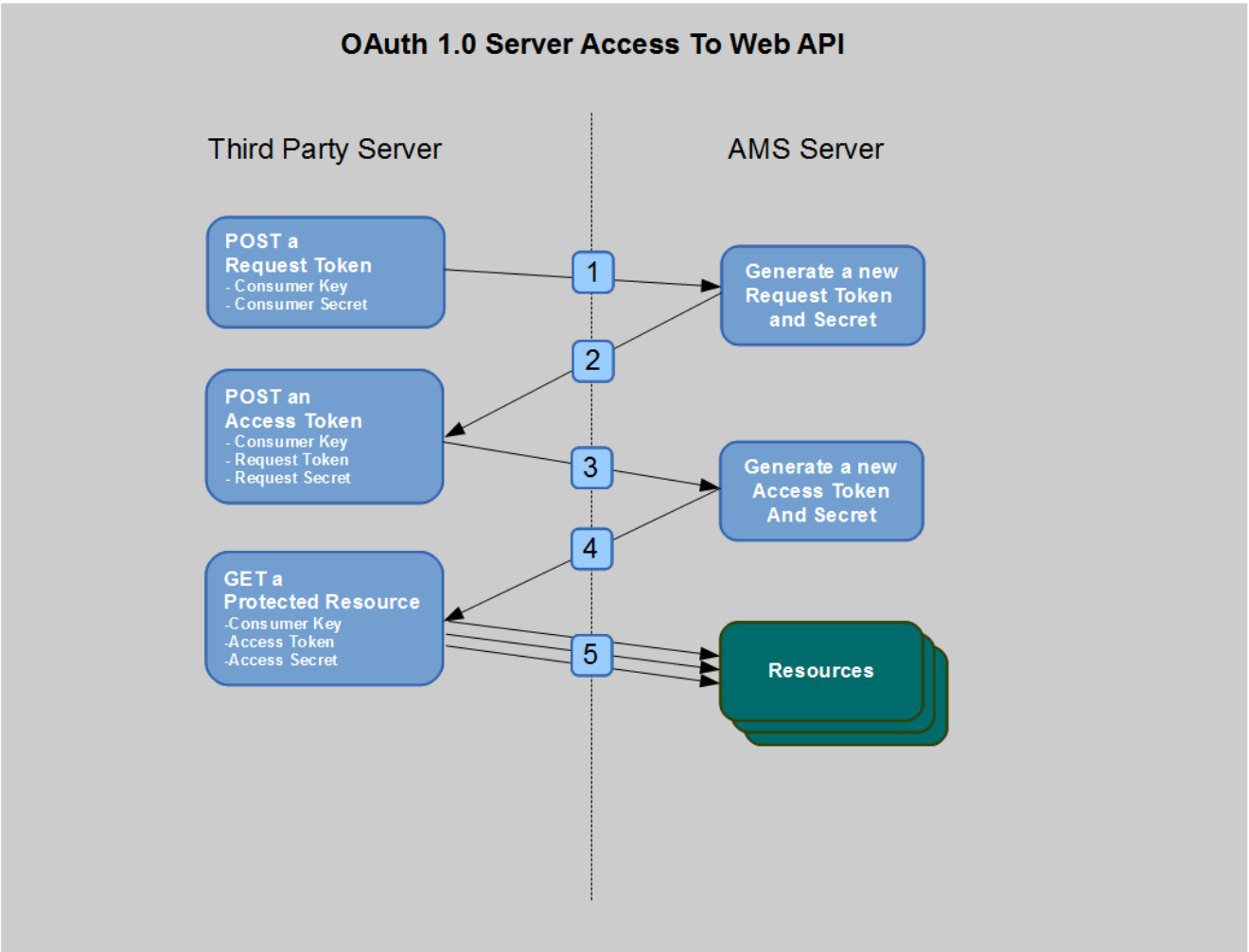
OAuth is an access control protocol that is a standardization of many well-established industry protocols. It is used by many companies such as Twitter, LinkedIn, Facebook, and TripIt. The goal of OAuth is to protect resources exposed by a service provider (AgileAssets Web API) and used by consumer applications on behalf of users.
With OAuth, an end user may grant access to its data to a service provider without providing its user credentials to the consumer application (Client flow).
For AgileAssets Web API, it means that a third-party application may access our services directly without any knowledge of our users (Server flow).

Server Flow (Two-legged OAuth)

This flow requires two identifiers in the authorization header:

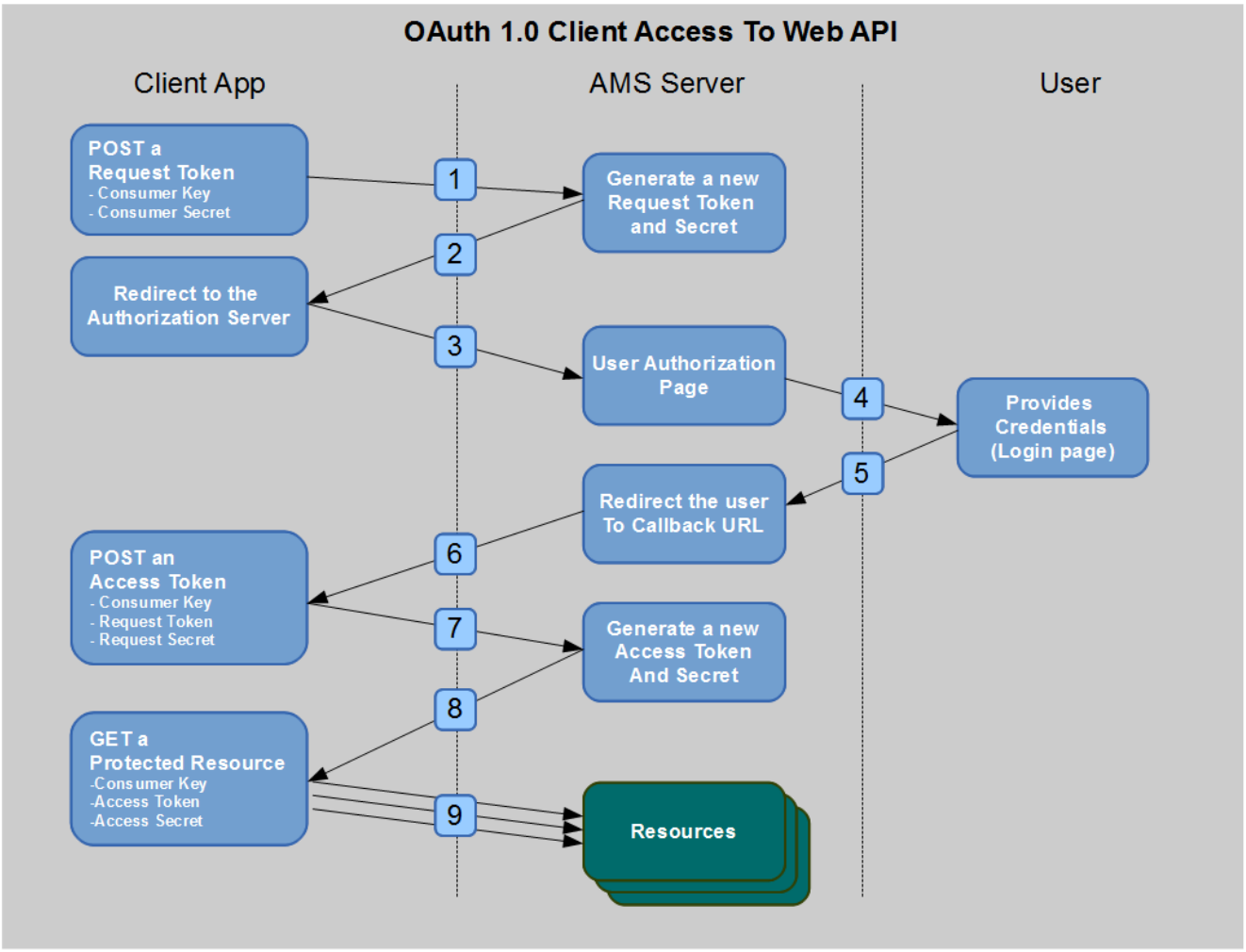
- The OAuth Access token granted for the Consumer Key
- The Consumer Secret

Optionally, the request can include an additional header field identifying the user account to modify. It is the responsibility of the caller to provide the user ID if the web service needs it.
The diagram below illustrates the Server Flow type of OAuth.



Client Flow (Three-legged OAuth)

This flow requires end-user participation to complete the request. The affected user must log in and confirm that the client application is allowed to access the API.
The diagram below illustrates the Client Flow type of OAuth.

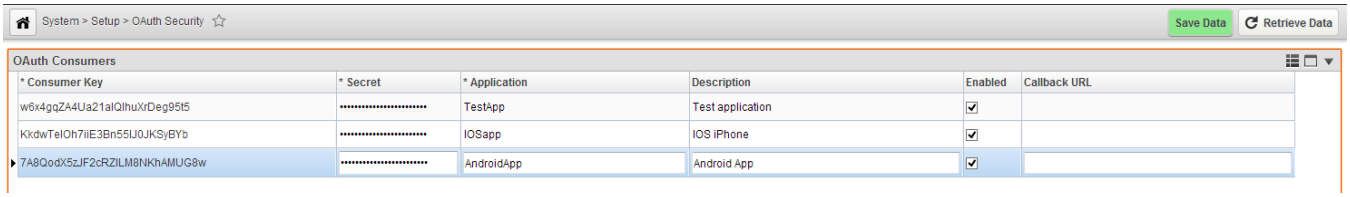


OAuth Configuration

Configuration is accomplished in three steps as described below.

Step 1: Add Consumer

The first step is to register a new consumer in the AgileAssets system in the OAuth Security window. (This window is typically launched from the Setup menu in the System module.) In this window, you enter the public key and the consumer secret that your client will use when making requests to the Web API as shown in the example below. After you have saved the information, ensure that OAuth authentication is enabled (that is, that a check mark appears in the Enabled column).



Note: The consumer secret is encrypted on the database side. For security reason the secret is never decrypted on the user-interface sided. When updating the secret, you must change it completely. If "Client Authorization Flow" is enabled, the client will be re-directed to an authorization page where the end user may give permission to the client application to log in.

Step 2: Configure the Application's web.xml

To activate API Security, you must ensure that the Init parameter com.sun.jersey.spi.container.ContainerRequestFilters is set for the RestService as shown in the example below.

Step 3: Configure the Client

The client will require the following information to make authentication requests to the Web API:

OAuth Config	Value
Request Token URL	WMS_BASE_URL + "/rest/v1/token/request"
Authorization URL	WMS_BASE_URL + "/Kernel/oauth_authorization.jsp""
Access Token URL	WMS_BASE_URL + "/rest/v1/token/access"
OAuth Signing Types	PLAINTEXT or HMAC-SHA1
Consumer Key	As configured in Step 1

Example of Two-legged OAuth

The following steps illustrate how to perform a two-legged OAuth authorization:

1. Obtain a request token by making a request via CURL:

```
curl -X POST --header 'Authorization: OAuth
oauth_consumer_key="1zN8HRxla7I2tlvbCaVXlKWsPS73",
oauth_signature="123q123Q%26",
oauth_nonce="0",
oauth_timestamp="1383847134",
oauth_signature_method="PLAINTEXT"'
 http://localhost:8080/ams_trunk/rest/v1/token/request
```

   The server responds as shown below:

```
{
"oauth_token": "RLrfjvSkm7hZGRkMxjXSFwLtp7rA",
"oauth_token_secret": "9DTljTDbvTKUYgmhh2hXWa7tTvQT",
"oauth_callback_confirmed": "true"
}
```

2. Obtain an access token by making a request via CURL:

```
curl -X POST -H 'Authorization: OAuth
oauth_token="RLrfjvSkm7hZGRkMxjXSFwLtp7rA",
oauth_consumer_key="1zN8HRxla7I2tlvbCaVXlKWsPS73",
oauth_signature="123q123Q%269DTljTDbvTKUYgmhh2hXWa7tTvQT",
oauth_nonce="07",
oauth_timestamp="1383876101",
oauth_signature_method="PLAINTEXT"'
http://localhost:8080/ams_trunk/rest/v1/token/access
```

   The server responds as shown below:

```
{
"oauth_token": "liWi1dmpbhA4xEneipQ1x0GvoTTY",
"oauth_token_secret": "HjRWOT74VRmF3x4Mw2ZZXb8112fv"
}
```

3. Access a resource (the person identified by the User ID JEFF) by making a request via CURL:

```
curl -X POST -H 'Authorization: OAuth
oauth_token="liWi1dmpbhA4xEneipQ1x0GvoTTY",
oauth_nonce="0",
oauth_signature_method="PLAINTEXT",
oauth_consumer_key="1zN8HRxla7I2tlvbCaVXlKWsPS73",
oauth_timestamp="1383876698",
oauth_signature="123q123Q%26HjRWOT74VRmF3x4Mw2ZZXb8112fv"'
http://localhost:8080/ams_trunk/rest/v1/user/JEFF
```

The server responds as shown below:


```
{
"USER_ID": "JEFF",
"ALLOW_LOGIN": 1,
"PASSWORD_START": "20080619110041",
"SEQ_PASSWORD": null,
"OWNER_ID": 1317,
"EMAIL": null,
"IS_ADMIN": 0,
"LEASE_OWNER_ID": null,
"DRIVER_LICENCE": null,
"ADD_AGENCY_CODE": null,
"CALLER_TEL": null,
"REMARKS": null,
"FIRST_NAME": null,
"LAST_NAME": null,
"MID_INITIAL": null,
"NAME": null,
"GROUP_CODE": null,
"UNSUCCESS_NUM": 0,
"SECURITY_QUESTIONS_ANSWERED": 0,
"PASSWORD_EXPIRED": 0,
"AD_USER_ID": "JEFF",
"USER_UPDATE": null,
"DATE_UPDATE": null
}
```


Example of Three-legged OAuth

The following steps illustrate how to perform a three-legged OAuth authorization:

1. Obtain a request token by making a request via CURL:

```
curl -X POST --header 'Authorization: OAuth
oauth_consumer_key="1zN8HRxla7I2tlvbCaVXlKWsPS73",
oauth_signature="123q123Q%26",
oauth_nonce="0",
oauth_timestamp="1383847134",
oauth_signature_method="PLAINTEXT"'
 http://localhost:8080/ams_trunk/rest/v1/token/request
```

The server responds as shown below:


```
{
"oauth_token": "RLrfjvSkm7hZGRkMxjXSFwLtp7rA",
"oauth_token_secret": "9DTljTDbvTKUYgmhh2hXWa7tTvQT",
"oauth_callback_confirmed": "true"
}
```


2. Obtain an authorization token. The request is re-directed to the provider to collect additional information. The request should contain output from the server's response in Step 1.
The server responds as shown below:


```
{
"oauth_verifier":"12345",
"oauth_token": "liWi1dmpbhA4xEneipQ1x0GvoTTY"
}
```

3. Obtain an access token by making a request via CURL:

```
curl -X POST -H 'Authorization: OAuth
oauth_token="RLrfjvSkm7hZGRkMxjXSFwLtp7rA",
oauth_consumer_key="1zN8HRxla7I2tlvbCaVXlKWsPS73",
oauth_signature="123q123Q%269DTljTDbvTKUYgmhh2hXWa7tTvQT",
oauth_nonce="07",
oauth_timestamp="1383876101",
oauth_verifier="12345",
oauth_signature_method="PLAINTEXT"'
http://localhost:8080/ams_trunk/rest/v1/token/access
```

The server responds as shown below:

```
{
"oauth_token": "liWi1dmpbhA4xEneipQ1x0GvoTTY",
"oauth_token_secret": "HjRWOT74VRmF3x4Mw2ZZXb8112fv"
}
```

4. Access a resource (the person identified by the User ID JEFF) by making a request via CURL:

```
{
"oauth_token": "liWi1dmpbhA4xEneipQ1x0GvoTTY",
"oauth_token_secret": "HjRWOT74VRmF3x4Mw2ZZXb8112fv"
}
```

The server responds as shown below:

```
{
"USER_ID": "JEFF",
"ALLOW_LOGIN": 1,
"PASSWORD_START": "20080619110041",
"SEQ_PASSWORD": null,
"OWNER_ID": 1317,
"EMAIL": null,
"IS_ADMIN": 0,
"LEASE_OWNER_ID": null,
"DRIVER_LICENCE": null,
"ADD_AGENCY_CODE": null,
"CALLER_TEL": null,
"REMARKS": null,
"FIRST_NAME": null,
"LAST_NAME": null,
"MID_INITIAL": null,
"NAME": null,
"GROUP_CODE": null,
"UNSUCCESS_NUM": 0,
"SECURITY_QUESTIONS_ANSWERED": 0,
"PASSWORD_EXPIRED": 0,
"AD_USER_ID": "JEFF",
"USER_UPDATE": null,
"DATE_UPDATE": null
}
```

Example of iPhone Authorizing

Shown below is an example of how to authorize an iPhone user via OAuth protocol:

```
- (NSError *)authorizeUser:(NSString *)userId inDomain:(NSString *)domain withPassword:(NSString *)password
        {
        NSString *entity = domain
                ? [NSString stringWithFormat:@"token/authorise/%@/domain/%@/password/%@", userId, domain, password ? password : @""]
                : [NSString stringWithFormat:@"token/authorise/%@/password/%@", userId, password ? password : @""];

        NSError *error = [[NSError new] autorelease];
        if ([self.restClient doPUT:entity withData:nil error:&error])
            return nil;

        return error;
        }
```

Once authorized, the iPhone user could, for example, obtain a specific work order ID via the following code:

```
- (bool)readLabor:(NSString *)workOrderId error:(NSError **)pError
    {
    NSData *d = [self.restClient doGET:[NSString stringWithFormat:@"workorder/%@/DayCard/labor", workOrderId]
                             withQueue:_qLabor
                                 error:pError];
    [_labor removeAllObjects];
    if (d)
        {
        NSArray *val = [self.restClient fetchedJsonObject:d error:pError];
        if (val)
            {
            [_labor overwriteWithRowCollection:val];
            _isLabor = true;
            }
        return true;
        }
    if (pError)
        [AAErrorInstance performSelectorOnMainThread:@selector(showErrorMessage:) withObject:*pError waitUntilDone:NO];
    return false;
    }
```