Predix

Implementing Security in Predix

Student Lab Guide

February 2016

GE Digital

Predix

Getting Started

This guide provides step-by-step instructions for lab exercises. Each lab corresponds to a topic covered in class and provides students with hands-on experience developing applications on the Predix platform.

Course Prerequisites:

- Introduction to Cloud Foundry for Predix
- Predix Fundamentals

Lab 1: Create UAA Service Instance Bind Service to Application

Learning Objectives

By the end of the lab, you will be able to:

- Create a UAA Service Instance and bind it to an application

Lab Exercises

Exercise 1: Learning Objectives

By the end of the lab, you will be able to:

- Create a UAA Service Instance and bind it to an application

Lab Exercises

- *Create a UAA Service Instance*, page 3

Exercise 1: Create a UAA Service Instance

Overview

In this exercise you will create an UAA service instance. You will bind your application to that instance.

Steps

1. Log into Cloud Foundry.

- Open a Terminal (Double click the Terminal icon on your desktop)
- In the Terminal run the command

  ```
  cf login
  ```

  - When prompted, enter your login ID, provided by your instructor
  - Enter your password, provided by your instructor
- You are prompted to select an org
  - Type the number with the selection for **Predix Training** and press **Enter**
- You are prompted to select a space

- Enter the number of the targeted space that your instructor provides

```
predix@localhost:~
File  Edit  View  Search  Terminal  Help
[predix@localhost ~]$ cf login
API endpoint: https://api.system.aws-usw02-pr.ice.predix.io

Email> student75

Password>
Authenticating...
OK

Targeted org Predix-Training

Select a space (or press enter to skip):
1. Training1
2. Training2
3. Training3

Space> 2
Targeted space Training2



API endpoint:   https://api.system.aws-usw02-pr.ice.predix.io
 (API version: 2.28.0)
User:           student75
Org:            Predix-Training
Space:          Training2
[predix@localhost ~]$
```

2. Verify the UAA service is available in the marketplace.

- In the Terminal run the command:

  `cf marketplace`

  - The **predix-uaa-training** service is listed along with its Plan name and description

```
service                    plans              description
logstash                   free               Logstash 1.4 serv
logstash-5                 free               Logstash 1.4 serv
p-rabbitmq                 standard           RabbitMQ is a rob
p-rabbitmq-35              standard           RabbitMQ is a rob
postgres                   shared, shared-nr  Reliable PostgrSQ
predix-acs                 Beta, Enterprise*  Use this service
predix-acs-training        Basic, Free        Design precise ac
predix-analytics-catalog   Beta, Enterprise*  Add analytics to
predix-analytics-runtime   Beta, Enterprise*  Use this service
predix-asset               Beta, Enterprise*  Create and store
predix-mobile              Beta, Enterprise*  Design, develop,
predix-timeseries          Beta, Enterprise*  Quickly and effic
predix-uaa                 Beta, Enterprise*  Use this service
predix-uaa-training        Free, Basic        Design precise me
predix-views               Beta, Enterprise*  Control layout an
redis-1                    shared-vm          Redis service to
riakcs                     developer          An S3-compatible

* These service plans have an associated cost. Creating a servic

TIP:  Use 'cf marketplace -s SERVICE' to view descriptions of in
```

3. Create a UAA service instance.

- In the Terminal run the command to create a service instance with the following syntax:

  `cf create-service predix-uaa-training <plan> <my_uaa_instance> -c '{"adminClientSecret":"<my_secret>"}'`

  - Replace **`<plan>`** with the plan name (e.g. beta, free)
  - Replace **`<my_uaa_instance>`** with an instance name of your choice
  - Replace **`<my_secret>`** with a password of your choice

  Sample command line for creating a service instance:

  `cf create-service predix-uaa-training Free Joni_uaa_instance -c '{"adminClientSecret":"training_secret"}'`

- Verify a status of **OK** is returned

```
[predix@localhost ~]$ cf create-service predix-uaa-training Free
 Joni_uaa_instance -c '{"adminClientSecret":"training_secret"}'
Creating service instance Joni_uaa_instance in org Predix-Traini
ng / space Training2 as student75...
OK
```

4. Bind a sample application to a UAA service instance.

- In the Terminal run the command:

  `cf bind-service <your_app_name> <my_uaa_instance>`

  - Replace **`<your_app_name>`** with **trainingsample_DoNotDelete**

    Note: This is a sample application created for you in your space

  - Replace **`<uaa_instance_name>`** with name of your instance

- A return status of **OK** indicates the application was successfully bound

```
[predix@localhost ~]$ cf bind-service trainingsample_DoNotDelete
 Joni_uaa_instance
Binding service Joni_uaa_instance to app trainingsample_DoNotDel
ete in org Predix-Training / space Training2 as student75...
OK
TIP: Use 'cf restage trainingsample_DoNotDelete' to ensure your
env variable changes take effect
```

Tip: To unbind your service, run this command:

```
cf unbind-service <your_app_name> <uaa_instance_name>
```

Lab 2: Create a Client and Users

Learning Objectives

By the end of the lab, you will be able to:

- Login as the administrator client
- Add a new client
- Create a user

Lab Exercises

Exercise 1: Fetch a UAA Token

Overview

In this exercise you will use the UAA Command Line Interface (uaac) to fetch a token from the UAA service instance created in the previous lab. The uaac has been installed on your DevBox.

Steps

1. Find your sample application name in the space.

- In the Terminal run the command:

  `cf a`

- Locate your application in the list and verify it has started

```
name                          requested state   instances
trainingsample_DoNotDelete    started           1/1
```

2. Retrieve your UAA instance details from the VCAP_SERVICES environment variable.

- In the Terminal, run the command:

  `cf env trainingsample_DoNotDelete`

- In the **VCAP_SERVICES** variable, locate the entry for your UAA service instance
  TIP: search for the name of your service instance (example: `Joni_uaa_instance`)

- Under the *credentials* section, copy the **uri** value for your service instance (copy the string <u>between</u> the quotes)


```
"predix-uaa-training": [
 {
  "credentials": {
   "issuerId": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.pr
   "uri": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-
   "zone": {
    "http-header-name": "X-Identity-Zone-Id",
    "http-header-value": "ad6e23ef-079a-4086-8d75-65c8bc0e7c38"
   }
  },
  "label": "predix-uaa-training",
  "name": "Joni_uaa_instance",
  "plan": "Free",
  "tags": []
 }
```

3. Specify your UAA instance as the intended target.

> ***Note:*** You must first *target* the uaac CLI tool to a specific UAA service instance, before you can run the CLI to view details about the service instance.

- In the Terminal run the command:

  `uaac target <uri>`

  - Replace <uri> with the uri copied from the output of the **`cf env`** command

```
[predix@localhost ~]$ uaac target https://ad6e23ef-079a-4086-8d7

Target: https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa-
```

4. Fetch a UAA token from your UAA instance.

- In the Terminal run the command to log in using the administrative client:

  `uaac token client get admin -s <my_secret>`

  Note: The *admin* client is the default client id that has all permissions

- Replace **`<my_secret>`** with the password you created when creating the service instance

  A successful fetch notice indicates you have retrieved the token

```
[predix@localhost ~]$ uaac token client get admin -s training_s
ecret

Successfully fetched token via client credentials grant.
Target: https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa
-training.run.aws-usw02-pr.ice.predix.io
Context: admin, from client admin
```

- You are now logged in as the administrative client

Note: Once we authenticate against the UAA service instance, a token is returned. Any subsequent authentications will now use this token.

5. Decrypt the token to view its contents.

- In the Terminal run the command:

  `uaac token decode`

```
[predix@localhost ~]$ uaac token decode

Note: no key given to validate token signature

  jti: cc181a70-32ff-493f-8081-26f61520b127
  sub: admin
  scope: clients.read
      zones.ad6e23ef-079a-4086-8d75-65c8bc0e7c38.admin
      clients.secret idps.write uaa.resource clients.write
      clients.admin idps.read scim.write scim.read
  client_id: admin
  cid: admin
  azp: admin
  grant_type: client_credentials
  rev_sig: 8a60c55b
  iat: 1454347929
  exp: 1454391129
  iss: https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa-
      training.run.aws-usw02-pr.ice.predix.io/oauth/token
  zid: ad6e23ef-079a-4086-8d75-65c8bc0e7c38
  aud: admin clients zones.ad6e23ef-079a-4086-8d75-65c8bc0e7c38
      idps uaa scim
```

- The token contains basic information including
  - authorities - a list of permissions when the token represents the client (application) itself
  - scope - a list of permissions that this client has on behalf of this user
  - client_id - unique name to the system for the client id
  - scope - a list of permissions that this client has on behalf of this user

Exercise 2: Adding a Client and Users to UAA

Overview

In this exercise you will create an OAuth2 client that authenticates users with a local UAA as an identity provider. When you create a UAA service instance, a default administrator account (admin client) is automatically generated that contains *all* permissions.

Steps

1. Create an OAuth2 client with a subset of admin permissions.

- In the Terminal, run the command:

  ```
  uaac client add -h
  ```

  This displays all parameters available to create a new OAuth client
- In the Terminal run the command (all one line):

  ```
  uaac client add <client_name> -s <my_secret>
  --authorized_grant_types "authorization_code client_credentials
  refresh_token password" --autoapprove openid --authorities
  "clients.read clients.write scim.write scim.read"
  ```

  Replace **`<client_name>`** with your client name
  - Replace **`<my_secret>`** with the password you created when creating the **UAA** service instance

Sample code:

```
uaac client add App_1_Client -s training_secret
--authorized_grant_types "authorization_code client_credentials
refresh_token" --autoapprove openid --authorities
"clients.read clients.write scim.write scim.read"
```

```
[predix@localhost ~]$ uaac client add App_1_Client -s training_
secret --authorized_grant_types "authorization_code client_cred
entials refresh_token" --autoapprove openid --authorities "clie
nts.read clients.write scim.read scim.write"
  scope: uaa.none
  client_id: App_1_Client
  resource_ids: none
  authorized_grant_types: authorization_code client_credentials
      refresh_token
  autoapprove:
  action: none
  authorities: clients.read clients.write scim.write scim.read
  lastmodified: 1454349152360
  id: App_1_Client
```

■

2. Fetch a token for the *new* OAuth2 client.

- In the Terminal, run the command:

  `uaac token client get <new_client> -s <my_secret>`

  - Replace **`<new_client>`** with the name of your new oauth2 client
  - Replace **`<my_secret>`** with the password you created when creating the uaa service instance

  Sample code: **`uaac token client get App_1_Client -s training_secret`**

- Verify the token was fetched successfully - you are now logged in as the *new* client

```
[predix@localhost ~]$ uaac token client get App_1_Client -s tra
ining_secret

Successfully fetched token via client credentials grant.
Target: https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa
-training.run.aws-usw02-pr.ice.predix.io
Context: app_1_client, from client App_1_Client
```

- Alternatively, you can exclude the -s portion of the command, and you will be prompted to enter the client secret. This is more secure since the password is not displayed as you login as App_1_Client.

3. Add a user to your UAA service instance.

Analysis: For applications accessing your UAA instance, you can create additional users with required scopes. You must be logged in as a client with the necessary authorities.

- In the Terminal, run the command:

  `uaac user add -h`

  This displays all parameters available for creating a new user
- In the Terminal run the command:

  `uaac user add <user_name> --emails <user_email> -p <user_password>`
  - Replace **`<user_name>`** with a value of your choice
  - Replace **`<user_email>`** with the email of the user (*e.g. user1@test.com*)
  - Replace **`<user_password>`** with a value of your choice
- Note down your user name and password: _____

```
[predix@localhost ~]$ uaac user add Wendy --emails wendy@test.c
om -p W3ndy
user account successfully added
```

4. Create the groups in your UAA instance.

- Give your user read and write privileges.

  `uaac group add scim.read`

  `uaac group add scim.write`

  Sample code:

```
[predix@localhost ~]$ uaac group add scim.read
  meta
    version: 0
    created: 2016-02-01T18:25:10.740Z
    lastmodified: 2016-02-01T18:25:10.740Z
  schemas: urn:scim:schemas:core:1.0
  id: 76795a56-cb93-456d-906b-72f36ea967ea
  displayname: scim.read
[predix@localhost ~]$ uaac group add scim.write
  meta
    version: 0
    created: 2016-02-01T18:25:21.586Z
    lastmodified: 2016-02-01T18:25:21.586Z
  schemas: urn:scim:schemas:core:1.0
  id: 73ce3227-0862-47d4-8e8c-9c49d5a34ba2
  displayname: scim.write
```

5. Add the new user to the required groups.

```
uaac member add scim.read <my-user>
uaac member add scim.write <my-user>
```

```
[predix@localhost ~]$ uaac member add scim.read Wendy
success
[predix@localhost ~]$ uaac member add scim.write Wendy
success
```

6. View user details.

- To view all users, run the command:

  `uaac users`

```
[predix@localhost ~]$ uaac users
  resources:
  -
    id: 0bfff9ee-8b79-45f0-b1a1-d16af5ce0dfa
    meta
      version: 0
      created: 2016-02-01T18:20:28.945Z
      lastmodified: 2016-02-01T18:20:28.945Z
    name
    emails:
    -
      value: wendy@test.com
      primary: false
    groups:
    -
      value: 76795a56-cb93-456d-906b-72f36ea967ea
      display: scim.read
      type: DIRECT
    -
      value: 73ce3227-0862-47d4-8e8c-9c49d5a34ba2
      display: scim.write
      type: DIRECT
    approvals:
    active: true
    verified: false
    origin: uaa
    schemas: urn:scim:schemas:core:1.0
    username: Wendy
    zoneid: ad6e23ef-079a-4086-8d75-65c8bc0e7c38
    passwordlastmodified: 2016-02-01T18:20:28.000Z
```

7. Authenticate to your UAA service instance as a user.

- Locate the UAA uri for your service instance

  In the Terminal run the command:

  `cf env <application_name>`

  Replace `<application_name>` with **`trainingsample_DoNotDelete`**

- In the **VCAP_SERVICES** variable, locate the entry for your UAA service instance

  Copy the **uri** value (this is the URL that you will use to access the service)

```
"predix-uaa-training": [
 {
  "credentials": {
   "issuerId": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.pr
   "uri": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-
   "zone": {
    "http-header-name": "X-Identity-Zone-Id",
    "http-header-value": "ad6e23ef-079a-4086-8d75-65c8bc0e7c38"
   }
  },
  "label": "predix-uaa-training",
  "name": "Joni_uaa_instance",
  "plan": "Free",
  "tags": []
 }
```

- Open a browser and paste the uri

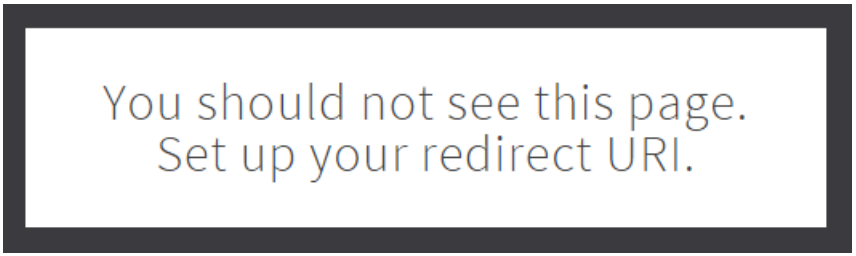
  Append the value **`/login`** to the uri

https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa-training.run.aws-usw02-pr.ice.predix.io/login

- At the Predix login screen, enter your **user** name (**not** email) and user password

Click **Sign in**

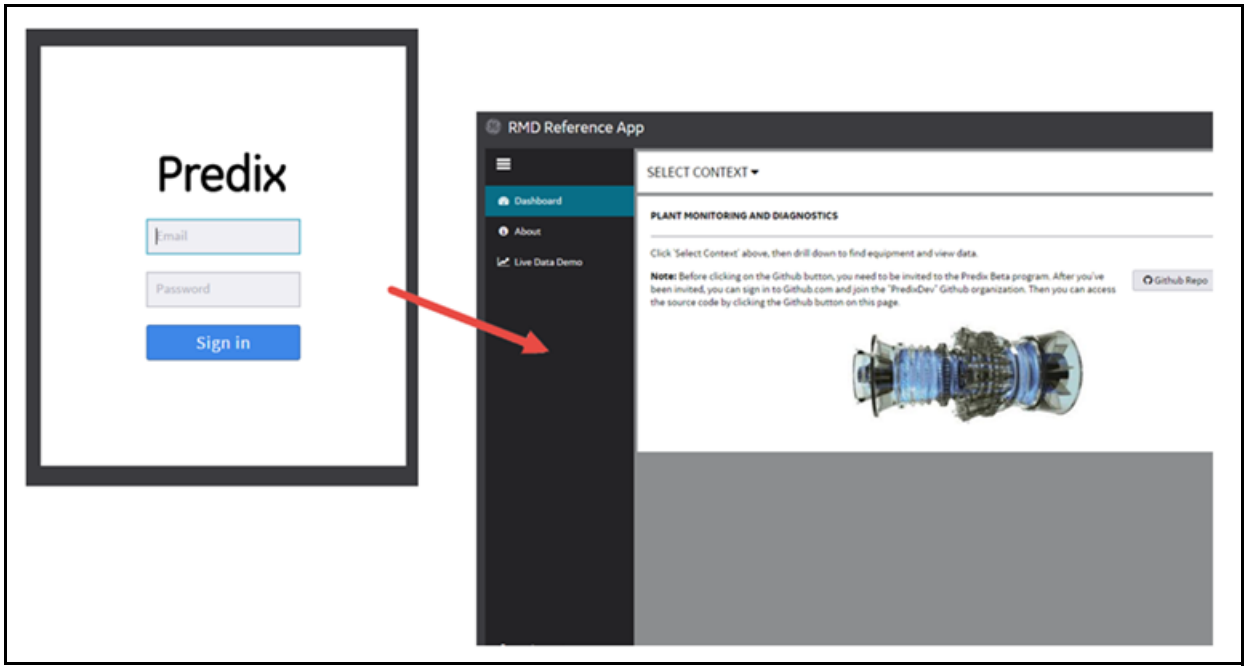
- The login is successful, but an error message is displayed:



Note: When creating the new client, we did not specify a redirect attribute. Users created by this client will not have a redirect attribute as well.

```
client add [name]                Add client registration
                                 --scope <list>
                                 --authorized_grant_types <list>
                                 --authorities <list>
                                 --access_token_validity <seconds>
                                 --refresh_token_validity <seconds>
                                 --redirect_uri <list>
                                 --autoapprove <list>
                                 --signup_redirect_url <url>
                                 --clone <other>, get default
                                 settings from other
                                 -s | --secret <secret>, client
                                 secret
                                 -i | --[no-]interactive,
                                 interactively verify all values
```

If the redirect_uri had been set you would have been directed to your application.



Lab 3: Authorize Using ACS

Learning Objectives

By the end of the lab, you will be able to:

- Create an ACS instance
- Bind your Application to the ACS instance
- Update an OAuth2 client to work with ACS
- Manage ACS User Access

Lab Exercises

Exercise 1: Create an ACS Instance

Overview

In this exercise you will create an ACS instance, bind your application to the instance.

Steps

1. Verify the UAA service is available in the marketplace.

- In the Terminal run the command:

  `cf marketplace`

  The **predix-acs-training** service is listed along with its Plan name and description

```
service                    plans                 description
logstash                   free                  Logstash 1.4 serv
logstash-5                 free                  Logstash 1.4 serv
p-rabbitmq                 standard              RabbitMQ is a rob
p-rabbitmq-35              standard              RabbitMQ is a rob
postgres                   shared, shared-nr     Reliable PostgrSQ
predix-acs                 Beta, Enterprise*     Use this service
predix-acs-training        Basic, Free           Design precise ac
predix-analytics-catalog   Beta, Enterprise*     Add analytics to
predix-analytics-runtime   Beta, Enterprise*     Use this service
predix-asset               Tiered                Create and store
predix-mobile              Beta, Enterprise*     Design, develop,
```

2. Create an ACS service instance.

- In the Terminal run the command to create a service instance with the following syntax:

```
cf create-service predix-acs-training <plan> <my_acs_instance>
-c '{"trustedIssuerIds":"<uaa_instance_issuerID>"}'
```

  - Replace `<plan>` with the plan name (e.g. Basic, Free)
  - Replace `<my_acs_instance>` with an instance name of your choice
  - Replace `<uaa_instance_issuerID>` is the **issuerID** (**not** the uri) of your UAA instance. This can be retrieved from the VCAP_SERVICES environment variable.

  Sample command line code for creating a service instance:

```
cf env trainingsample_DoNotDelete
```

```
  "predix-uaa-training": [
   {
    "credentials": {
     "issuerId": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.p
redix-uaa-training.run.aws-usw02-pr.ice.predix.io/oauth/token",
     "uri": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix
-uaa-training.run.aws-usw02-pr.ice.predix.io",
     "zone": {
      "http-header-name": "X-Identity-Zone-Id",
      "http-header-value": "ad6e23ef-079a-4086-8d75-65c8bc0e7c38
"
     }
    },
    "label": "predix-uaa-training",
    "name": "Joni_uaa_instance",
    "plan": "Free",
    "tags": []
   }
```

```
cf create-service predix-acs-training Free Joni_acs_instance -c
'{"trustedIssuerIds":"https://ad6e23ef-079a-4086-8d75-65c8bc0e7
c38.predix-uaa-training.run.aws-usw02-pr.ice.predix.io/oath/tok
en"}'
```

- Verify a status of OK is returned

```
[predix@localhost ~]$ cf create-service predix-acs-training Free
 Joni_acs_instance -c '{"trustedIssuerIds":"https://ad6e23ef-079
a-4086-8d75-65c8bc0e7c38.predix-uaa-training.run.aws-usw02-pr.ic
e.predix.io/oauth/token"}'
Creating service instance Joni_acs_instance in org Predix-Traini
ng / space Training2 as student75...
OK
```

Exercise 2: Bind your Application to the ACS instance

Overview

You must bind your application to your ACS instance to provision its connection details in the VCAP_SERVICES environment variable.

Steps

1. Bind your application to the new ACS instance.

- In the Terminal execute the command:

  `cf bind-service <your_app_name> <acs_instance_name>`

  - Replace **`<your_app_name>`** with **`trainingsample_DoNotDelete`**
  - Replace **`<acs_instance_name>`** with your ACS instance name

  Sample command line code for binding a service instance to an Application:

  `cf bind-service trainingsample_DoNotDelete Joni_acs_instance`

```
[predix@localhost ~]$ cf bind-service trainingsample_DoNotDelete
 Joni_acs_instance
Binding service Joni_acs_instance to app trainingsample_DoNotDel
ete in org Predix-Training / space Training2 as student75...
OK
TIP: Use 'cf restage trainingsample_DoNotDelete' to ensure your
env variable changes take effect
```

- Verify the binding:

  `cf env <your_app_name>`

  Sample command line code for viewing the VCAP_SERVICES environment variable:

  `cf env trainingsample_DoNotDelete`


```
"predix-acs-training": [
 {
  "credentials": {
   "uri": "https://predix-acs-training.run.aws-usw02-pr.ice.pre
   "zone": {
    "http-header-name": "Predix-Zone-Id",
    "http-header-value": "ccff1702-ef11-4769-a527-deade4c917b0",
    "oauth-scope": "predix-acs-training.zones.ccff1702-ef11-476
   }
  },
  "label": "predix-acs-training",
  "name": "Joni_acs_instance",
  "plan": "Free",
  "tags": []
 }
```


Notice the label is **predix-acs-training** and the name is **Joni_acs_instance**

Exercise 3: Update an OAuth2 Client to Work with ACS

Overview

To enable applications to manage policies and attributes using ACS, you need to update your OAuth2 client with the required OAuth2 scopes and authorities. In this exercise, you will establish your OAuth2 client to handle Policy Management Services. This will handle tokens sent by the application or client.

Steps

1. Specify your UAA instance as the intended target, if needed. If you are already targeted to the UAA Training instance, then skip to the next step.

- In the Terminal execute the command:

```
uaac target <uaa_instance_url>
```

- Replace **`<uaa_instance_url>`** with your uri, retrieved from the **VCAP_SERVICES** environment variable

```
"predix-uaa-training": [
 {
  "credentials": {
   "issuerId": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.pre
   "uri": "https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-u
   "zone": {
    "http-header-name": "X-Identity-Zone-Id",
    "http-header-value": "ad6e23ef-079a-4086-8d75-65c8bc0e7c38"
   }
  },
  "label": "predix-uaa-training",
  "name": "Joni_uaa_instance",
  "plan": "Free",
  "tags": []
 }
```

Sample command line code for targeting a service instance:

`uaac target https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa-training.run.aws-usw02-pr.ice.predix.io`

```
[predix@localhost ~]$ uaac target https://ad6e23ef-079a-4086-8d75

Target: https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa-t
Context: app_1_client, from client App_1_Client
```

2. Ensure you are logged into UAAC using the client you created in Lab 2.

- Run the command:

  `uaac token client get <client name>`

  Enter your UAA instance password when prompted.

  Sample command line code for getting a client token:

  `uaac token client get App_1_Client`

  Password

  `training_secret`

```
[predix@localhost ~]$ uaac token client get App_1_Client
Client secret:  ***************

Successfully fetched token via client credentials grant.
Target: https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa-
Context: app_1_client, from client App_1_Client
```

- Run the command:

uaac context

```
[predix@localhost ~]$ uaac context

[13]*[https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa

  [1]*[app_1_client]
      client_id: App_1_Client
      access_token: eyJhbGciOiJSUzI1NiJ9.eyJqdGkiOiI3OTk3YTQy
JlYWQiLCJjbGllbnRzLndyaXRlIiwic2NpbS53cml0ZSIsInNjaW0ucmVhZCJ
CJncmFudF90eXBlIjoiY2xpZW50X2NyZWRlbnRpYWxzIiwicmV2X3NpZyI6Ij
YS00MDg2LThkNzUtNjVjOGJjMGU3YzM4LnByZWRpeC11YWEtdHJhaW5pbmcuc
kNzUtNjVjOGJjMGU3YzM4IiwiYXVkIjpbIkFwcF8xX0NsaWVudCIsImNsaWVu
ghG1_sAuzhvg-w-_tgAvuZJPoU40xK0ZddnuM-T3ExlGB0k9b8vmaLpt5wZiS
RS0-o11ZhoN1FJwPkfJz1kRs8EnKKcU4bgZIC_yhWKGXKG51yBbxsTmCSz19s
      token_type: bearer
      expires_in: 43199
      scope: clients.read clients.write scim.write scim.read
      jti: 7997a42c-18f1-4539-b40c-ec5af931d0d9
```

- Note the scope for App_1_Client.

3. Update the OAuth2 client with the authorities required for ACS.

- Login as admin to make these changes. Run the command:

  `uaac token client get admin`

  Specify the **`<client secret>`** when prompted.

```
[predix@localhost ~]$ uaac token client get admin
Client secret:  ***************

Successfully fetched token via client credentials grant.
Target: https://ad6e23ef-079a-4086-8d75-65c8bc0e7c38.predix-uaa-
training.run.aws-usw02-pr.ice.predix.io
Context: admin, from client admin
```

- Update the OAuth2 client to match the required ACS authorities.
  - Run the command **`uaac clients`**, and **copy** zones from admin, which will allow the client to create groups:

    `zones.<xxx>.admin`

    Sample result:

```
[predix@localhost ~]$ uaac clients
  admin
     scope: uaa.none
     resource_ids: none
     authorized_grant_types: client_credentials
     autoapprove:
     action: none
     authorities: clients.read
         zones.ad6e23ef-079a-4086-8d75-65c8bc0e7c38.admin
         clients.secret idps.write uaa.resource clients.write
         clients.admin idps.read scim.write scim.read
     lastmodified: 1454346221583
```

- Originally you created your application client authorities with the original list:

  `clients.read`
  `clients.write`
  `scim.write`
  `scim.read`

- Add to the list the ACS requirements:

  `acs.policies.read`
  `acs.policies.write`
  `acs.attributes.read`
  `acs.attributes.write`

  `<oauth-scope>` `(from the application env VCAP-SERVICES)`

  To find the value of the authority for **oauth-scope** in your **VCAP-SERVICES**

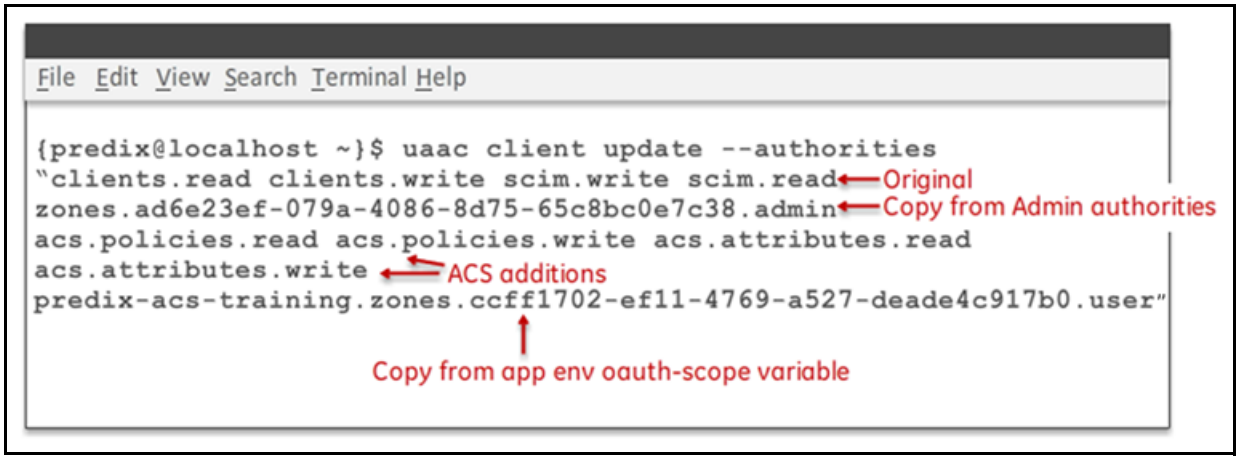
  ```
  cf env trainingsample_DoNotDelete
  ```

  Sample return value:


```
"predix-acs-training": [
 {
  "credentials": {
   "uri": "https://predix-acs-training.run.aws-usw02-pr.ice.pred
   "zone": {
    "http-header-name": "Predix-Zone-Id",
    "http-header-value": "ccff1702-ef11-4769-a527-deade4c917b0",
    "oauth-scope": "predix-acs-training.zones.ccff1702-ef11-4769
   }
  },
  "label": "predix-acs-training",
  "name": "Joni_acs_instance",
  "plan": "Free",
  "tags": []
 }
```

Sample code:

```
uaac client update --authorities "clients.read clients.write
scim.write scim.read
zones.ad6e23ef-079a-4086-8d75-65c8bc0e7c38.admin
acs.policies.read acs.policies.write acs.attributes.read
acs.attributes.write
predix-acs-training.zones.ccff1702-ef11-4769-a527-deade4c917b0.
user"
```



- When prompted, enter the client name you created

  Sample response:

  App_1_Client

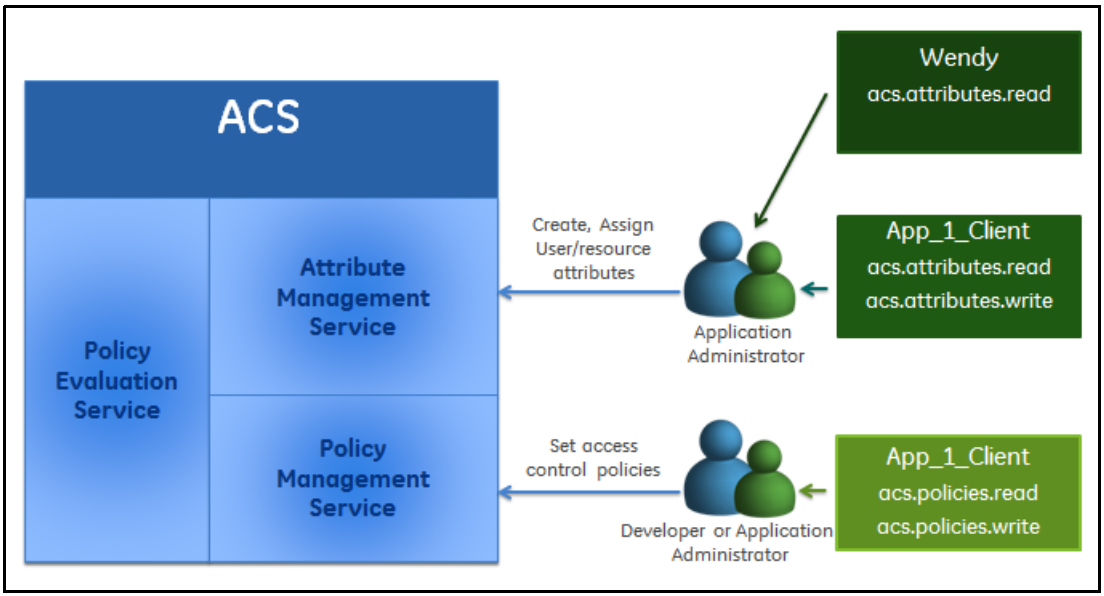
Sample results for updating the OAuth2 client:

```
App_1_Client
  scope: uaa.none
  resource_ids: none
  authorized_grant_types: authorization_code
      client_credentials password refresh_token
  autoapprove:
  action: none
  authorities: clients.read acs.policies.read
      zones.ad6e23ef-079a-4086-8d75-65c8bc0e7c38.admin
      acs.policies.write acs.attributes.read clients.write
      acs.attributes.write
      predix-acs-training.zones.ccff1702-ef11-4769-a527-deade
      4c917b0.user scim.write scim.read
  lastmodified: 1454690761664
```

Exercise 4: Manage ACS User Access

Overview

App_1_Client will manage the resources for Attribute Management Services, and the account and permissions will need to be set up. You created user Wendy in a previous lab. When Wendy logs in and creates a report, she will need read permissions.



Steps

1. Create the groups required for ACS in UAA.

You can use Admin to create groups, although as a best practice, you should use your application client to manage policies and attributes in your application. You updated your application client in Exercise 3 to enable that client to manage ACS attributes and policies.

- Ensure your UAA instance is the intended target

  ```
  uaac target <uri>
  ```

- Logon as the administrative client

  ```
  uaac token client get <app client>
  ```

  Specify the **`<client_secret>`** when prompted

- Create the groups required for ACS in UAA:

  ```
  uaac group add acs.policies.read
  uaac group add acs.policies.write
  uaac group add acs.attributes.read
  uaac group add acs.attributes.write
  ```

Sample command line code to create groups

```
[predix@localhost ~]$ uaac group add acs.policies.read
  meta
    version: 0
    created: 2016-02-01T22:17:44.573Z
    lastmodified: 2016-02-01T22:17:44.573Z
  schemas: urn:scim:schemas:core:1.0
  id: 83d7e29e-262a-4899-880f-36895b8a7164
  displayname: acs.policies.read
[predix@localhost ~]$ uaac group add acs.policies.write
  meta
    version: 0
    created: 2016-02-01T22:18:01.524Z
    lastmodified: 2016-02-01T22:18:01.524Z
  schemas: urn:scim:schemas:core:1.0
  id: c0419f89-a762-44ae-b929-c064f38ca41d
  displayname: acs.policies.write
```

```
[predix@localhost ~]$ uaac group add acs.attributes.read
  meta
    version: 0
    created: 2016-02-01T22:18:15.594Z
    lastmodified: 2016-02-01T22:18:15.594Z
  schemas: urn:scim:schemas:core:1.0
  id: 45694018-720e-440e-83dc-288c12deac57
  displayname: acs.attributes.read
[predix@localhost ~]$ uaac group add acs.attributes.write
  meta
    version: 0
    created: 2016-02-01T22:18:31.765Z
    lastmodified: 2016-02-01T22:18:31.765Z
  schemas: urn:scim:schemas:core:1.0
  id: 5cfc25ba-d98a-4af8-b0ce-1b4b8f2877d0
  displayname: acs.attributes.write
```

- The last command requires the environment variable oauth-scope from VCAP_SERVICES

  `cf env trainingsample_DoNotDelete`

  Sample command line code

```
"predix-acs-training": [
 {
  "credentials": {
   "uri": "https://predix-acs-training.run.aws-usw02-pr.ice.pred
   "zone": {
    "http-header-name": "Predix-Zone-Id",
    "http-header-value": "ccff1702-ef11-4769-a527-deade4c917b0",
    "oauth-scope": "predix-acs-training.zones.ccff1702-ef11-4769
   }
  },
  "label": "predix-acs-training",
  "name": "Joni_acs_instance",
  "plan": "Free",
  "tags": []
 }
```

- Copy the **oauth-scope** value and run the command:

  `uaac group add predix-acs.zones.<acs_instance_guid>.user`

  Where **`acs_instance_guid`** is generated in **VCAP_SERVICES** environment variable as **`oauth-scope`**

  Sample command line code to create groups

```
[predix@localhost ~]$ uaac group add predix-acs-training.zones.c
cff1702-ef11-4769-a527-deade4c917b0.user
  meta
    version: 0
    created: 2016-02-01T22:24:59.003Z
    lastmodified: 2016-02-01T22:24:59.003Z
  schemas: urn:scim:schemas:core:1.0
  id: 3b304244-2614-4c20-b6e8-0ada124aefda
  displayname: predix-acs-training.zones.ccff1702-ef11-4769-a527
      -deade4c917b0.user
```

2. To use the Policy Management service the user/client must authenticate using a JSON Web Token (JWT) bearer token that includes the acs.policies.read scope for reading the policies or the acs.policies.write scope for writing the policies. To use the Attribute Management service, the user/client must authenticate using a JWT that includes the acs.attributes.read and the acs.attributes.write scope.

- Create a user to add to your ACS groups.

  uaac user add <user_name> -p <user_password> --emails <user email>

  Sample command line code to add user:

```
[predix@localhost ~]$ uaac user add Jaime -p J@mi3 --emails
jamie@test.com
user account successfully added
```

- Assign membership to the required scope.

  uaac member add acs.policies.read <user_name>
  uaac member add acs.policies.write <user_name>
  uaac member add acs.attributes.read <user_name>
  uaac member add acs.attributes.write <user_name>
  uaac member add predix-acs.zones.<acs_instance_guid>.user <user_name>

Sample command line code to assign membership to the required scope:

```
[predix@localhost ~]$ uaac member add acs.attributes.read Jaime
success
[predix@localhost ~]$ uaac member add acs.attributes.write Jaim
e
success
```

```
[predix@localhost ~]$ uaac member add predix-acs-training.zones
.ccff1702-ef11-4769-a527-deade4c917b0.user Jaime
success
```

Jaime will now be able to authenticate through ACS Policy Management, Attribute Management, and Policy Evaluation services.