

ExtraCredit – RentTrack Integration

BuildCreditModule API Security - Setup Clients and Users

Contents

**[Register a new Client for accessing Build Credit Module APIs](#_Toc25856627)** [3](#_Toc25856627)

[**Register a new User for accessing Build Credit Module APIs** 4](#_Toc25856628)

[**BCrypt Password Generation using the Demo App** 5](#_Toc25856629)

[**Generate BCrypt Encoded text using the Encode Endpoint** 5](#_Toc25856630)

[**Validate BCrypt Encoded text using the Decode Endpoint** 5](#_Toc25856631)

# **Register a new Client for accessing Build Credit Module APIs**

Login to the BCM Database Server instance (Dev/Test/Stage/Prod as per the need). The database login ID used should have read & write access to the objects in ‘rtcustomer’ schema.

Use the following SQL, replace the values highlighted in Yellow accordingly (example provided below), and run it.

INSERT INTO  
 rtcustomer.oauth\_client\_details (  
 client\_id, resource\_ids, client\_secret, scope, authorized\_grant\_types, web\_server\_redirect\_uri, authorities, access\_token\_validity, refresh\_token\_validity, additional\_information, autoapprove  
 )

VALUES (  
 '<unique-client-id-string>', 'oauth2-resource', '<BCryptPassword>', 'read,write', 'authorization\_code,check\_token,refresh\_token,password', NULL, '<list\_of\_comma\_separated\_roles\_prefixed\_with\_**ROLE\_**>', <numeric\_access\_token\_validity\_duration\_in\_seconds>, <numeric\_refresh\_token\_validity\_duration\_in\_seconds>, NULL, NULL  
)

Example:

INSERT INTO  
 rtcustomer.oauth\_client\_details (  
 client\_id, resource\_ids, client\_secret, scope, authorized\_grant\_types, web\_server\_redirect\_uri, authorities, access\_token\_validity, refresh\_token\_validity, additional\_information, autoapprove  
 )   
VALUES (  
 'client', 'oauth2-resource', '$2a$10$Gy5adJIkNWHr9SMjUc4mWO5KwVk.aHYeNVUnth73a064VH0feWAU6', 'read,write', 'authorization\_code,check\_token,refresh\_token,password', NULL, 'ROLE\_CLIENT', 900, 21600, NULL, NULL  
)

**NOTE:** Only the ‘**bcm-web**’ module uses OAuth at present, which requires both Client and User registrations. The other module that exposes secured RESTFul APIs is ‘**bcm-jobs**’. The Jobs module doesn’t use OAuth, and hence, only User registration is required there, no need to register Clients.

# **Register a new User for accessing Build Credit Module APIs**

Login to the BCM Database Server instance (Dev/Test/Stage/Prod as per the need). The database login ID used should have read & write access to the objects in ‘rtcustomer’ schema.

Use the following SQL, replace the values highlighted in Yellow accordingly (example provided below), and run it.

INSERT INTO   
 rtcustomer.oauth\_user(id, email, password, role)   
VALUES   
 (<unique\_numeric\_id\_value>, '<unique\_user\_email@domain.com>', '<BCryptPassword>', '<list\_of\_comma\_separated\_roles\_prefixed\_with\_**ROLE\_**>');

Example:

INSERT INTO   
 rtcustomer.oauth\_user(id, email, password, role)   
VALUES   
 (10, 'rsrsp-bcmclient@progrexion.com', '$2a$10$Gy5adJIkNWHr9SMjUc4mWO5KwVk.aHYeNVUnth73a064VH0feWAU6', 'ROLE\_RSRSP');

# **BCrypt Password Generation using the Demo App**

BCM Web/OAuth security configuration uses BCryptPassword Encoder for handling the password/secret of Clients/Users. When registering a new Client or User by following the above mentioned steps, you would need to provide the secret/password as a BCrypt encoded String to be stored in the database table. To do this, any Java program that uses an instance of **org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder** and its **encode(**“plain-text-string”**)** method may be employed.

Right now, we are using a utility end-point provided by our EXTRACREDIT-BCM-AWSDEMO application hosted in Heroku, for getting BCrypt encoded strings. Steps to use this utility endpoint are noted below.   
**Note:** BCrypt uses a random Salt generated internally, and also, you most probably won’t get the same encoded-text twice for a given plain-text input. Also, the encoded string is basically a ‘hash’, so the encoded text can’t (normally) be ‘decrypted’ back to the original plain-text.

## **Generate BCrypt Encoded text using the Encode Endpoint**

Use a REST client tool such as Postman to hit the endpoint using the details from the example given below (inputs are highlighted in Yellow):

**Method:** POST  
**URL:** <https://extracredit-bcm-awsdemo-dev.herokuapp.com/encode>  
**Headers:** Content-Type: application/json  
**Request Body:**   
{ "in": "plain-text-password" }  
**Response Body:**   
{ "in": "plain-text-password", "out": "$2a$10$Gy5adJIkNWHr9SMjUc4mWO5KwVk.aHYeNVUnth73a064VH0feWAU6" }

## **Validate BCrypt Encoded text using the Decode Endpoint**

Use a REST client tool such as Postman to hit the endpoint using the details from the example given below (inputs are highlighted in Yellow):

**Method:** POST   
**URL:** <https://extracredit-bcm-awsdemo-dev.herokuapp.com/decode>  
**Headers:** Content-Type: application/json  
**Request Body:** { "plainPass": "plain-text-password",   
"encodedPass": "$2a$10$Gy5adJIkNWHr9SMjUc4mWO5KwVk.aHYeNVUnth73a064VH0feWAU6"}  
**Response Body:** { "plainPass": "plain-text-password",   
"encodedPass": "$2a$10$Gy5adJIkNWHr9SMjUc4mWO5KwVk.aHYeNVUnth73a064VH0feWAU6", "result": "matching" }