

My Estes Profile Administrator Service Outline

8/31/2018

Change History

| Document Version | Date | Changed by | Description |
| --- | --- | --- | --- |
| 1.0 | 8/31/2018 | Ataur Rahman | Initial version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

Contents

[1. Introduction 1](#_Toc523489778)

[1.1 Purpose 1](#_Toc523489779)

[1.2 Scope 1](#_Toc523489780)

[1.3 Overview 1](#_Toc523489781)

[2. Current App 2](#_Toc523489782)

[3. Services 2](#_Toc523489783)

[3.1 Profile Information 2](#_Toc523489784)

[3.1.1 Get User Profile Information 2](#_Toc523489785)

[3.2 Profile Management 4](#_Toc523489786)

[3.2.1 Change Username 4](#_Toc523489787)

[3.2.2 Change Email Address 5](#_Toc523489788)

[3.2.3 Change Password 6](#_Toc523489789)

[3.2.4 Change Email Update Preference 7](#_Toc523489790)

[3.3 Sub Account Management 8](#_Toc523489791)

[3.3.1 Create Account 9](#_Toc523489792)

[3.3.2 Edit User Access 13](#_Toc523489793)

[3.3.3 Change User Password 16](#_Toc523489794)

[3.3.4 Enable/Disable User 18](#_Toc523489795)

[3.3.5 Search & Listing 20](#_Toc523489796)

[4. Profile Database 24](#_Toc523489797)

[4.1 Tables 24](#_Toc523489798)

[4.1.1 ESTESRTGY2.QNP150 24](#_Toc523489799)

[4.1.2 ESTESRTGY2.QNP230 24](#_Toc523489800)

[4.1.3 ESTESRTGY2.QNP302 24](#_Toc523489801)

[4.2 SPROC 24](#_Toc523489802)

[4.2.1 FBPGMS.SP\_UPDUSERNAME 24](#_Toc523489803)

[4.2.2 FBPGMS.SP\_UPDPASSWORD 24](#_Toc523489804)

[4.2.3 FBPGMS.SP\_UPDEMAILADDR 24](#_Toc523489805)

[4.2.4 FBPGMS.SP\_UPDUPDATEPREF 25](#_Toc523489806)

[4.2.5 FBPGMS.SP\_CREATESUBACCOUNT 25](#_Toc523489807)

[4.3 SQL 26](#_Toc523489808)

# Introduction

The My Estes Profile Administrator app provides the services to manage profile services

## Purpose

Existing functionality should be identified to determine what services are needed before coding starts.

## Scope

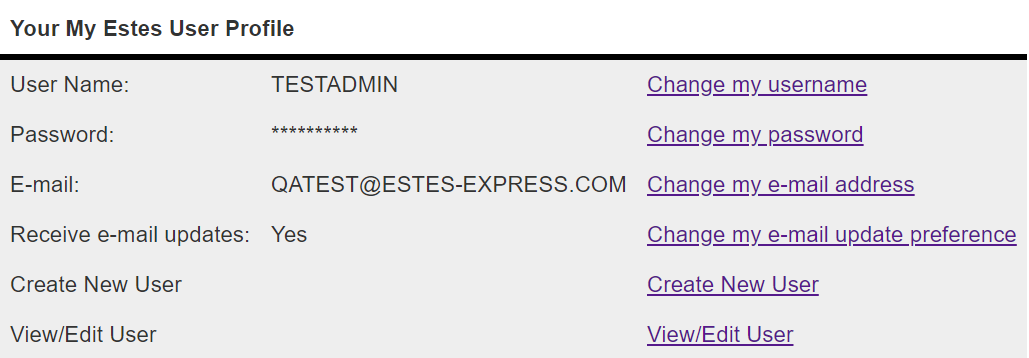
The scope of this document covers only the My Estes Profile Administrator app.

## Overview

The Profile Administrator app allows My Estes users can change username, password, email address, email update preference. Users who have the privilege to create sub account, can create new user, enable/disable user, change password, list & search the user, manage application permissions for the sub accounts.

# Current App

The app presents a menu of options available to manage profile.



There is no service needed to emulate the menu.

# Services

A number of services are needed to emulate the existing UI.

## Profile Information

### Get User Profile Information

* + User needs to be authenticated to access this service.
  + **GET http://qa.estesinternal.com/api/myestes/Profile/v1.0/me**
    - It provides the authenticated User Profile Information.
    - Send Authorization in header to get access
    - Sample Successful Response for DEMO user:
    - {  
       "firstName": "DEMO",  
       "lastName": "DEMO",  
       "username": "DEMO",  
       "email": "LNASH@ESTES-EXPRESS.COM",  
       "phone": "(804) 353-1900",  
       "companyName": "DEMO",  
       "accountType": "L",  
       "accountCode": "0212345",  
       "createdDate": "20180829",  
       "status": "ENABLED",  
       "preference": "Y"  
      }
    - Response DTO
    - @Data  
      @ApiModel(description="User Profile")  
      public class Profile  
      {  
       @ApiModelProperty(position=1, notes="First Name")  
       private String firstName;  
        
       @ApiModelProperty(position=2, notes="Last Name")   
       private String lastName;  
        
       @ApiModelProperty(position=3, notes="Username")  
       private String username;  
        
       @ApiModelProperty(position=4, notes="Email Address")  
       private String email;  
        
       @ApiModelProperty(position=5, notes="Phone Number")  
       private String phone;  
        
       @ApiModelProperty(position=6, notes="Company Name")  
       private String companyName;  
        
       @ApiModelProperty(position=7, notes="Account Type")  
       private String accountType;  
        
       @ApiModelProperty(position=8, notes="Account Code")  
       private String accountCode;  
        
       @ApiModelProperty(position=9, notes="Created Date")  
       private String createdDate;  
        
       @ApiModelProperty(position=10, notes="User Status: ENABLED/DISABLED")  
       private UserStatus status;  
        
       @ApiModelProperty(position=11, notes="Email Update Preference. Y for Yes, N for No")  
       private String preference;  
        
      }

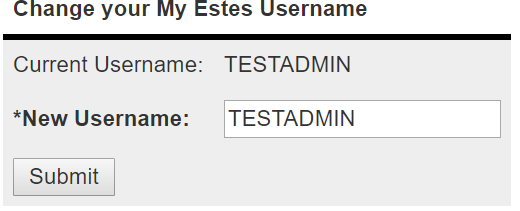
## Profile Management

User needs to be authenticated to access these services. User should have **EDITPROF** scope. In another words, user should have permission to access the Edit Profile Information. Send Authorization in header to get access these services.

### Change Username

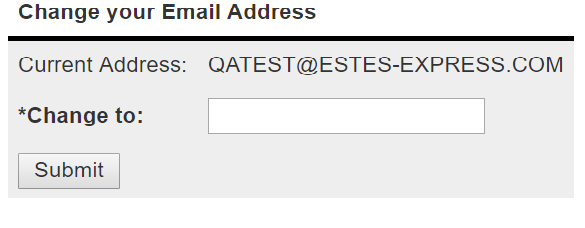
Changing Username is very critical services. It replaces the old username by new username in all the tables the username has. After changing the username, user should be logged out. Because, username in token is not changed after changing the username. That’s why, after changing username, if a user tries to access further resource, it will be failed.

To solve this problem, after changing username, I just forced the user logged out automatically. & I did this.



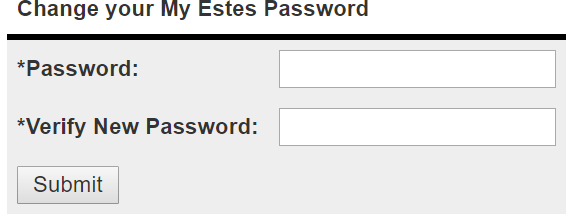
* + **POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/change\_username**
    - It changes the username of authenticated user.
    - Username should not be null.
    - Username should have minimum 5 characters & maximum 10 characters
    - Sample Request Body
    - {  
       "username" : "NEW\_USERNAME"  
      }
    - Sample Successful Response
    - {  
       "success" : true,  
       "message" : " Successful Message"  
      }
    - Sample Error Response
    - {  
       "error" : "BAD\_REQUEST",  
       "error\_description" : "Error Response Message"  
      }
    - DTO for Username
    - @ApiModel(description="This model will be used for updating username")  
      @Data  
      public class Username {  
       @NotNull  
       @Size(min=5,max=10,message="Username should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(name="Username",notes="Username should have minimum 5 & maximum 10 characters")  
       private String username;  
      }

### Change Email Address



* + **POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/change\_email**
    - It changes the email address of authenticated user.
    - Email Address should not be null & should be valid email address.
    - Sample Request Body
    - {  
       "email" : "new\_email@mail.com"  
      }
    - Sample Successful Response
    - {  
       "success" : true,  
       "message" : " Successful Message"  
      }
    - Sample Error Response
    - {  
       "error" : "BAD\_REQUEST",  
       "error\_description" : "Error Response Message"  
      }
    - DTO for Email Address
    - @Data  
      @ApiModel(description="This model will be used to update user email address")  
      public class EmailAddress {  
       @NotNull  
       @ApiModelProperty(name="Email Address",notes="User new email Address")  
       private String email;  
      }

### Change Password

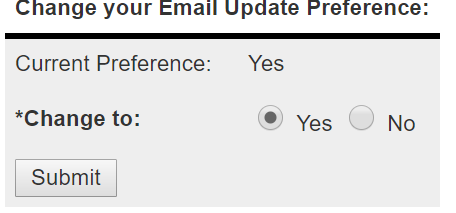


* + **POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/change\_password**
    - It changes the password for authenticated user.
    - Password should not be null
    - Password should have minimum 5 characters & maximum 10 characters
    - Password & Confirm Password should be identical
    - Sample Request Body
    - {  
       "password" : "new\_password",  
       "confirmPassword": "confirmPassword"  
      }
    - Sample Successful Response
    - {  
       "success" : true,  
       "message" : " Successful Message"  
      }
    - Sample Error Response

{  
 "error" : "BAD\_REQUEST",  
 "error\_description" : "Error Response Message"  
}

* + - DTO for Password
    - @Data  
      @FieldsMatch(field = "password", fieldMatch = "confirmPassword", message = "Passwords do not match!")  
      @ApiModel(description="Password model will be used to change password")  
      public class Password {  
        
       @NotNull  
       @Size(min=5, max=10, message = "Password should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(position=1,notes="Password should have minimum 5 & maximum 10 characters. Password should match the confirm password")  
       private String password;  
        
       @NotNull  
       @Size(min =5, max=10, message = "Confirm Password should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(position=2,name="Confirm Password",notes="Confirm Password should have minimum 5 & maximum 10 characters. Confirm password should match the Password")  
       private String confirmPassword;  
        
      }

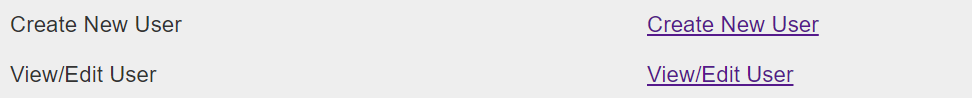
### Change Email Update Preference



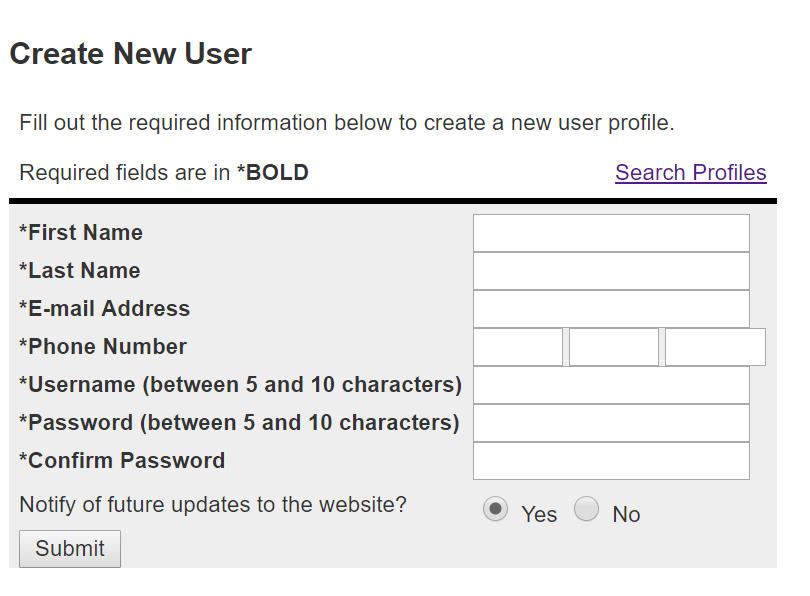
* + **http://qa.estesinternal.com/api/myestes/Profile/v1.0/change\_email\_preference**
    - It changes the email preference of authenticated user.
    - Email Preference should not be null & should be either '**Y**' or '**N**'.
    - Sample Request Body
    - {  
       "preference" : "Y"  
      }
    - Sample Successful Response
    - {  
       "success" : true,  
       "message" : " Successful Message"  
      }
    - Sample Error Response
    - {  
       "error" : "BAD\_REQUEST",  
       "error\_description" : "Error Response Message"  
      }
    - DTO for Email Preference
    - @Data  
      @ApiModel(description="This model will be used for updating email preference")  
      @NotNull(message="Email Preference can not be null")  
      public class EmailPreference {  
        
       @NotNull(message="Preference can not be null")  
       @ApiModelProperty(name="Email Preference",notes="Email Preference can have two value 'Y' or 'N'")  
       private String preference;  
        
       @AssertTrue(message="Preference can be either 'Y' or 'N'")  
       public boolean isPreference() {  
       return preference!=null && ( preference.trim().equalsIgnoreCase("Y") || preference.trim().equalsIgnoreCase("N"));  
       }  
      }

## Sub Account Management

User needs to be authenticated to access these services. User should have **EDITPROF** as well as **ADMINSUBS** scopes. In another words, user should have permission to access the Sub Account Management. Send Authorization in header to get access these services

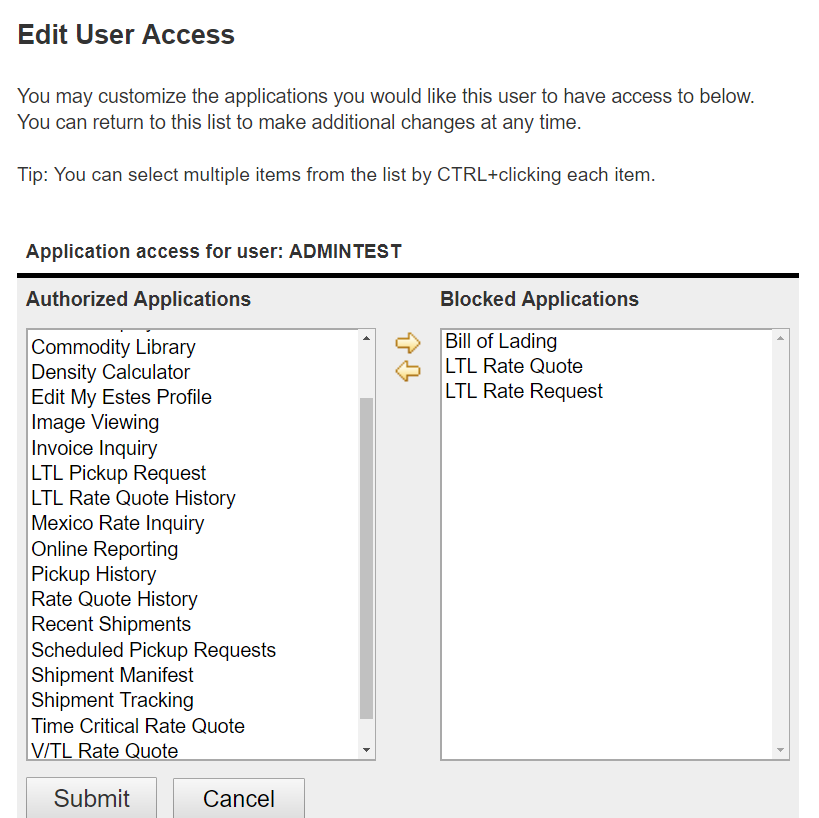


### Create Account



* + **POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/users**
    - It create a new sub account
    - Fields:
      * **firstName**:
        + First Name can not be null. Maximum 25 characters are allowed.
      * **lastName**:
        + Last Name can not be null. Maximum 25 characters are allowed.
      * **email**:
        + Should be valid email Address. Above Email Pattern is being used to validate email address.
      * **phone**:
        + Should be valid phone.
        + Sample Valid phone numbers are (804) 335 83659 , (804)-455-667, (804) 457-678 etc.
      * **username**:
        + username can not be null.
        + Mimimum 5 characters are required.
        + Maximum 10 characters are allowed
        + username should be unique
      * **password**:
        + Password should have minimum 5 characters & maximum 10 characters
      * **confirmPassword**:
        + Password & Confirm Password should be identical
      * **notify**:
        + Value can be either '**Y**' or '**N**'
    - Sample \_\_@RequestBody\_\_:
    - {  
       "firstName": "User First Name",  
       "lastName": "User Last Name",  
       "email": "user.email@test.com",  
       "phone": "(804) 336-8359",  
       "username": "username",  
       "password": "password",  
       "confirmPassword": "password",  
       "notify": "Y"  
      }
    - Request DTO:
    - @Data  
      @FieldsMatch(field = "password", fieldMatch = "confirmPassword", message = "Passwords do not match!")  
      @ApiModel(description="To create Sub Account")  
      public class User {  
        
       @NotNull  
       @Size(min=1, max=25)  
       @ApiModelProperty(position=1, name="firstName",notes="First Name")  
       private String firstName;  
        
       @NotNull  
       @Size(min=1,max=25)  
       @ApiModelProperty(position=2, name="lastName",notes="First Name")  
       private String lastName;  
        
       @NotNull  
       @ApiModelProperty(position=5, name="email",notes="Email Address")  
       private String email;  
        
       @NotNull  
       @ApiModelProperty(position=6, name="phone",notes="Phone Number")  
       private String phone;  
        
       @NotNull  
       @Size(min=5,max=10,message="Username should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(position=7,name="Username",notes="Username should have minimum 5 & maximum 10 characters")  
       private String username;  
       @NotNull(message = "Password can't be null")  
       @Size(min=5, max=10, message = "Password should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(position=8,notes="Password should have minimum 5 & maximum 10 characters. Password should match the confirm password")  
       private String password;  
        
       @NotNull  
       @Size(min =5, max=10, message = "Confirm Password should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(position=9,name="Confirm Password",notes="Confirm Password should have minimum 5 & maximum 10 characters. Confirm password should match the Password")  
       private String confirmPassword;  
        
       @NotNull  
       @ApiModelProperty(position=10, name="notify",notes="Notify can be either 'Y' or 'N")  
       private String notify;  
        
       @AssertTrue(message="Notify can be either 'Y' or 'N'")  
       public boolean isNotify() {  
       return notify!=null && ( notify.trim().equalsIgnoreCase("Y") || notify.trim().equalsIgnoreCase("N"));  
       }  
        
       public String getAreaCode(){  
       return phone.replaceAll("\\D+", "").substring(0, 3);  
       }  
        
       public String getExchange(){  
       return phone.replaceAll("\\D+", "").substring(3, 6);  
       }  
        
       public String getLastFour(){  
       return phone.replaceAll("\\D+", "").substring(6,10);  
       }  
        
      }

### Edit User Access



#### Get List of User Authorized Apps

* + **GET http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/authorized\_apps**
    - It provides the list of user authorized apps.
    - Request Parameters:
      * **username**: username can not be null (**Required**)
    - Sample Request: GET http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/authorized\_apps?username=DEMO
    - Sample Response:
    - [  
       {  
       "name": "EBG10O101",  
       "category": "G05",  
       "description": "Address Book"  
       },  
       {  
       "name": "BOL100",  
       "category": "Y01",  
       "description": "Bill of Lading"  
       },  
       {  
       "name": "CLAIMSFILE",  
       "category": "D03",  
       "description": "Claims Filing"  
       },  
       {  
       "name": "CLAIMIN",  
       "category": "D01",  
       "description": "Claims Inquiry"  
       },  
      ]

#### User Blocked Apps Management

After creation of any sub account, sub account gets all the privileges the admin user have except **AMINSUBS** Admin can block/unblock a user from any specific applications.

The following services does the block/unblock application for a user

##### Get List of User Blocked Apps

* + - **GET http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/blocked\_apps**
      * It provides the list of blocked applications for a user.
      * Request Parameters:
        + **username**: username can not be null (**Required**)
      * Sample Request: GET http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/blocked\_apps?username=DEMO
      * Sample Response:
      * [  
         {  
         "name": "EBG10O101",  
         "category": "G05",  
         "description": "Address Book"  
         },  
         {  
         "name": "BOL100",  
         "category": "Y01",  
         "description": "Bill of Lading"  
         },  
         {  
         "name": "CLAIMSFILE",  
         "category": "D03",  
         "description": "Claims Filing"  
         },  
         {  
         "name": "CLAIMIN",  
         "category": "D01",  
         "description": "Claims Inquiry"  
         },  
        ]

##### Add Apps to the User Blocked List

* + - **POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/blocked\_apps**
      * Add list of Apps to the blocked list of Applications for a user.
      * Request Parameters:
        + **username**: username can not be null (**Required**)
      * Request Body:
        + List of **appName**
        + Sample Values :
        + ["EBG10O101","BOL100"]

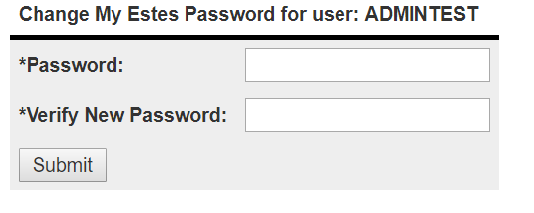
##### Remove Apps from User Blocked List

* + - **DELETE http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/blocked\_apps**
      * Delete list of Apps from the blocked list of Applications for a user.
      * Request Parameters:
        + **username**: username can not be null (**Required**)
      * Request Body:
        + List of **appName**
        + Sample Values :
        + ["EBG10O101","BOL100"]

##### Remove/Clear Blocked List

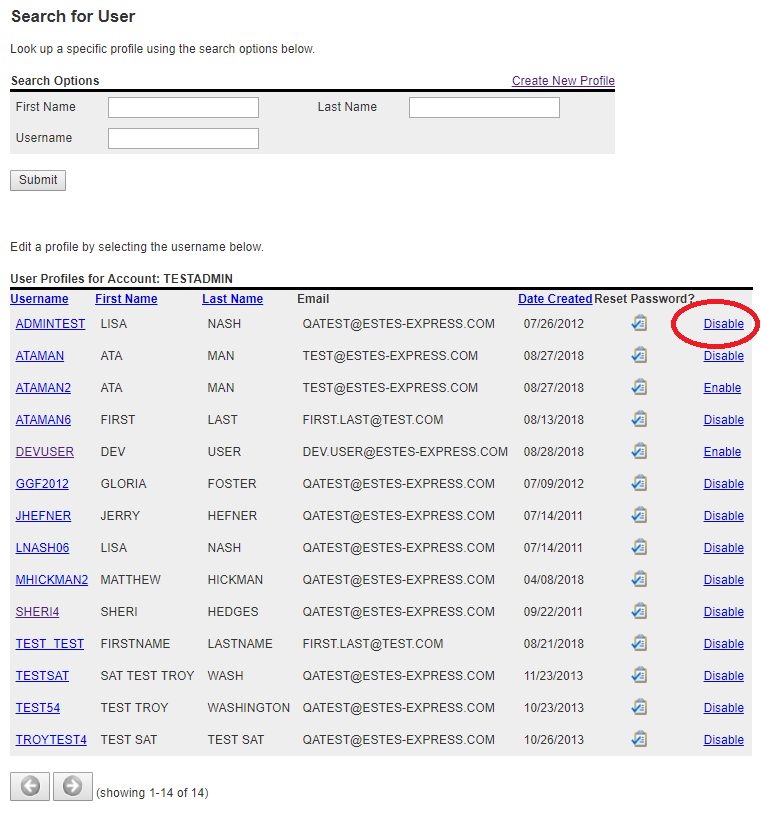
* + - **DELETE http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/blocked\_apps/all**
      * Delete/Clear the blocked list of Applications for a user.
      * Request Parameters:
        + **username**: username can not be null (**Required**)

### Change User Password



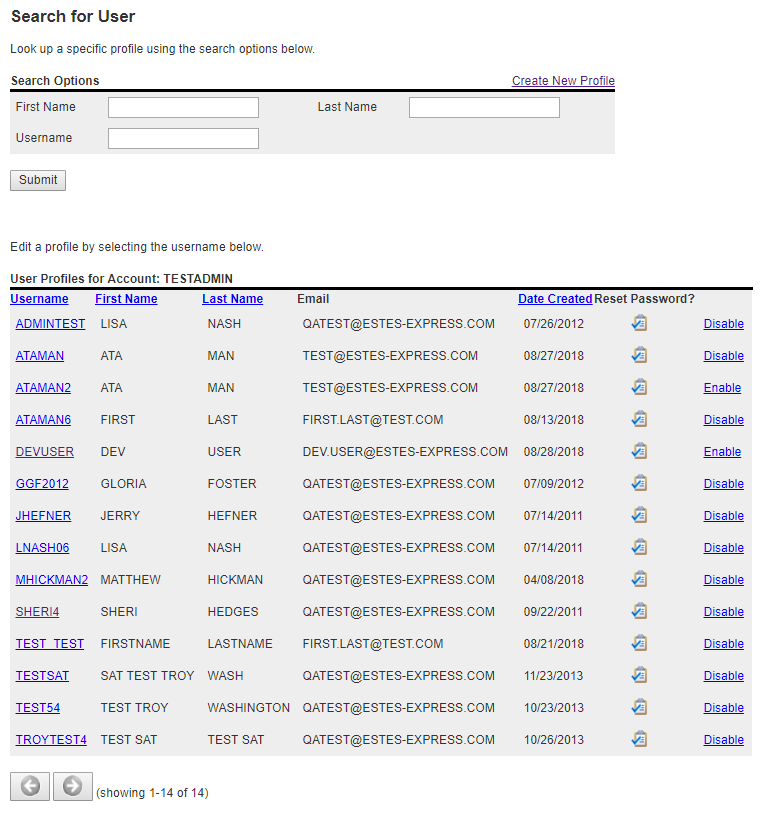
* + **POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/change\_password**
    - It changes the password for authenticated user.
    - Password should not be null
    - Password should have minimum 5 characters & maximum 10 characters
    - Password & Confirm Password should be identical
    - Sample Request Body
    - {  
       "password" : "new\_password",  
       "confirmPassword": "confirmPassword"  
      }
    - Sample Successful Response
    - {  
       "success" : true,  
       "message" : " Successful Message"  
      }
    - Sample Error Response
    - {  
       "error" : "BAD\_REQUEST",  
       "error\_description" : "Error Response Message"  
      }
    - DTO for Password
    - @Data  
      @FieldsMatch(field = "password", fieldMatch = "confirmPassword", message = "Passwords do not match!")  
      @ApiModel(description="Password model will be used to change password")  
      public class Password {  
        
       @NotNull  
       @Size(min=5, max=10, message = "Password should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(position=1,notes="Password should have minimum 5 & maximum 10 characters. Password should match the confirm password")  
       private String password;  
        
       @NotNull  
       @Size(min =5, max=10, message = "Confirm Password should have minimum 5 & maximum 10 characters")  
       @ApiModelProperty(position=2,name="Confirm Password",notes="Confirm Password should have minimum 5 & maximum 10 characters. Confirm password should match the Password")  
       private String confirmPassword;  
        
      }

### Enable/Disable User



* + **POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/change\_status**
    - It enables/disables the sub account.
    - **Request Parameters**:
      * **status**:
        + It can not be null
        + It is the new status of User
        + Value can be either **ENABLED** or **DISABLED**
        + **EANBLED** is used to enable the user & **DISABLED** is used to disable the user.
      * **username**:
        + It can not be null
        + The child Username
    - Sample Request With Query String :
      * POST http://qa.estesinternal.com/api/myestes/Profile/v1.0/users/change\_status?status=ENABLED&username=DEMO
    - Sample Successful Response
    - {  
       "success" : true,  
       "message" : " Successful Message"  
      }
    - Sample Error Response
    - {  
       "error" : "BAD\_REQUEST",  
       "error\_description" : "Error Response Message"  
      }

### Search & Listing



* + **GET http://qa.estesinternal.com/api/myestes/Profile/v1.0/users**
    - It provides the user list (sub account list) created by the authenticated user
    - The following **request parameters** can be passed to get advanced results.
      * **firstName** - To search by first name (**optional**)
      * **lastName** - To search by last name (**optional**)
      * **username** - To search by username (**optional**)
      * **q** - To search any string that will match to **firstName**,**lastName**,**username** and **email** (**optional**)
      * **rpp** - Results Per Page. Deafult value is **10**. (**optional**)
      * **page** Page no. (**optional**)
        + Deafult value is **1**.
        + You need to pass the page value for next pages.
        + Page value can not be less than **1** and greater than **totalPages**
      * **sortBy** - Results are sorted by this column. (**optional**)
        + Value can be any of the following:

**firstName**,**lastName**,**username**,**email**,**phone**,**companyName**,**accountCode**,**createdDate**,**status**

* + - * + Default value is **createdDate**.
      * **type** - Sort type. (**optional**)
        + Value can be either **asc** or **desc** .
        + default value is **asc**
    - Successful Response:
      * DTO Fields:
        + **page** - Current Page No
        + **totalResults** - Total Results without any filtration by above **q**,**firstName**,**lastName**,**username**.
        + **totalPages** - Total Page No without any filtration by above **q**,**firstName**,**lastName**,**username**.
        + **filtratedTotalResults** - Total Results after Filtering by above **q**,**firstName**,**lastName**,**username**.
        + **filtratedTotalPages** Total Page No after Filtering by above **q**,**firstName**,**lastName**,**username**.
        + **rpp** - Results Per Page
        + **results** List of Profile
        + **payload** Search Parameters if you sent.
      * **DTO**:
        + Profile Data:
        + @Data  
          @ApiModel(description="List of User Profile")  
          public class ProfileData {  
           @ApiModelProperty(position=1, notes="Current Page No")  
           private int page;  
           @ApiModelProperty(position=2, notes="Total Number of Page")  
           private int totalPages;  
           @ApiModelProperty(position=3, notes="Total Results")  
           private int totalResults;  
           @ApiModelProperty(position=4, notes="Total Number of Pages After Filtration")  
           private int filtratedTotalPages;  
           @ApiModelProperty(position=5, notes="Total Results after Filtration")  
           private int filtratedTotalResults;  
           @ApiModelProperty(position=6, notes="Results Per Page")  
           private int rpp;  
           @ApiModelProperty(position=7, notes="List of User Profile")  
           private List<Profile> results;  
           @ApiModelProperty(position=8, notes="Payload")  
           private Map<String, Object> payload;  
          }
        + Profile:
        + @Data  
          @ApiModel(description="User Profile")  
          public class Profile  
          {  
           @ApiModelProperty(position=1, notes="First Name")  
           private String firstName;  
           @ApiModelProperty(position=2, notes="Last Name")   
           private String lastName;  
           @ApiModelProperty(position=3, notes="Username")  
           private String username;  
           @ApiModelProperty(position=4, notes="Email Address")  
           private String email;  
           @ApiModelProperty(position=5, notes="Phone Number")  
           private String phone;  
           @ApiModelProperty(position=6, notes="Company Name")  
           private String companyName;  
           @ApiModelProperty(position=7, notes="Account Type")  
           private String accountType;  
           @ApiModelProperty(position=8, notes="Account Code")  
           private String accountCode;  
           @ApiModelProperty(position=9, notes="Created Date")  
           private String createdDate;  
           @ApiModelProperty(position=10, notes="User Status: ENABLED/DISABLED")  
           private UserStatus status;  
           @ApiModelProperty(position=11, notes="Email Update Preference. Y for Yes, N for No")  
           private String preference;  
          }
      * Json Response for user TESTADMIN with rpp=2:
      * {  
         "page": 1,  
         "totalPages": 7,  
         "totalResults": 14,  
         "filtratedTotalPages": 7,  
         "filtratedTotalResults": 14,  
         "rpp": 2,  
         "results": [  
         {  
         "firstName": "JERRY",  
         "lastName": "HEFNER",  
         "username": "JHEFNER",  
         "email": "JHEFNER@ESTES-EXPRESS.COM",  
         "phone": "(804) 353-1900",  
         "companyName": "DIAMOND CRYSTAL BRANDS",  
         "accountType": null,  
         "accountCode": "5023958",  
         "createdDate": "20110714",  
         "status": "ENABLED",  
         "preference": "N"  
         },  
         {  
         "firstName": "LISA",  
         "lastName": "NASH",  
         "username": "LNASH06",  
         "email": "LNASH@ESTES-EXPRESS.COM",  
         "phone": "(804) 353-1900",  
         "companyName": "DIAMOND CRYSTAL BRANDS",  
         "accountType": null,  
         "accountCode": "5023958",  
         "createdDate": "20110714",  
         "status": "ENABLED",  
         "preference": "Y"  
         }  
         ],  
         "payload": {  
         "q": null,  
         "firstName": null,  
         "lastName": null,  
         "sortBy": "createdDate",  
         "type": "asc",  
         "username": null  
         }  
        }

# Profile Database

## Tables

### ESTESRTGY2.QNP150

### ESTESRTGY2.QNP230

### ESTESRTGY2.QNP302

## SPROC

The following Store Procedures are being used:

### FBPGMS.SP\_UPDUSERNAME

To update username, call this store procedure

* CREATE PROCEDURE FBPGMS.SP\_UPDUSERNAME (   
   IN USRNAME CHAR(10) ,   
   IN NEWUSERNAME CHAR(10) ,   
   OUT "ERROR" CHAR(7)  
  )

### FBPGMS.SP\_UPDPASSWORD

To change password, call the store procedure

* CREATE PROCEDURE FBPGMS.SP\_UPDPASSWORD (   
   IN USRNAME CHAR(10) ,   
   IN NEWPWRD CHAR(10) ,   
   IN VERIFYPWRD CHAR(10) ,   
   OUT "ERROR" CHAR(7)  
  )

### FBPGMS.SP\_UPDEMAILADDR

To update email address, call the store procedure

* CREATE PROCEDURE FBPGMS.SP\_UPDEMAILADDR (   
   IN USRNAME CHAR(10) ,   
   IN EMAIL CHAR(50) ,   
   OUT "ERROR" CHAR(7)  
  )

### FBPGMS.SP\_UPDUPDATEPREF

To update email preference, call the store procedure

* CREATE PROCEDURE FBPGMS.SP\_UPDUPDATEPREF (   
   IN USRNAME CHAR(10) ,   
   IN UPDATE\_FLG CHAR(1) ,   
   OUT "ERROR" CHAR(7)  
  )

### FBPGMS.SP\_CREATESUBACCOUNT

To create sub account, call the following store procedure

* CREATE PROCEDURE FBPGMS.SP\_CREATESUBACCOUNT (   
   IN SIGNED\_IN\_USER CHAR(10) ,   
   IN FIRST\_NAME CHAR(25) ,   
   OUT FIRST\_NAME\_ERR CHAR(7) ,   
   IN LAST\_NAME CHAR(25) ,   
   OUT LAST\_NAME\_ERR CHAR(7) ,   
   IN CO\_NAME CHAR(50) ,   
   OUT CO\_NAME\_ERR CHAR(7) ,   
   IN ACCT\_CODE CHAR(7) ,   
   OUT ACCT\_CODE\_ERR CHAR(7) ,   
   IN ACCT\_REP CHAR(60) ,   
   OUT ACCT\_REP\_ERR CHAR(7) ,   
   IN EMAIL CHAR(50) ,   
   OUT EMAIL\_ERR CHAR(7) ,   
   IN AREA\_CODE CHAR(3) ,   
   OUT AREA\_CODE\_ERR CHAR(7) ,   
   IN EXCHANGE CHAR(3) ,   
   OUT EXCHANGE\_ERR CHAR(7) ,   
   IN LAST\_FOUR CHAR(4) ,   
   OUT LAST\_FOUR\_ERR CHAR(7) ,   
   IN USER\_NAME CHAR(10) ,   
   OUT USER\_NAME\_ERR CHAR(7) ,   
   IN "PASSWORD" CHAR(10) ,   
   OUT PASSWORD\_ERR CHAR(7) ,   
   IN PW\_CONFIRM CHAR(10) ,   
   OUT PW\_CONFIRM\_ERR CHAR(7) ,   
   IN NOTIFY CHAR(1) ,   
   OUT NOTIFY\_ERR CHAR(7) ,   
   OUT "ERROR" CHAR(7)  
  );

## SQL

* + To get Profile information, the following query is being used.
    - Params:
      * **QSUN** : signed in user's **username**
  + select qrfnam as firstName,  
    qrlnam as lastName,  
    QSACT# as accountCode,  
    CONCAT(CONCAT( CONCAT('(',qrpnac),CONCAT(') ', qrpnfp)), CONCAT('-',qrpnlp)) as phone,  
    QRCNAM as companyName,  
    QRUN as username,  
    qrem1 as email,  
    qrwebu as preference,  
    QSRDAT as createdDate,  
    CASE WHEN QZREFC IS NULL THEN 'ENABLED' ELSE 'DISABLED' END AS status  
    from estesrtgy2.qnp150  
    LEFT JOIN estesrtgy2.qnp230 ON QSUN=QRUN  
    LEFT JOIN ( select QZREFC from estesrtgy2.qnp302 where QZBLTP = 1 and QZREFT = 1) as t   
    ON QZREFC=QSUN  
    WHERE QSUN=?;
  + To retrieve list of Users Created By Signed In User
    - Params:
      * **QSEM2** : Signen In User's **username**
  + select qrfnam as firstName,  
    qrlnam as lastName,  
    QSACT# as accountCode,  
    CONCAT(CONCAT( CONCAT('(',qrpnac),CONCAT(') ', qrpnfp)), CONCAT('-',qrpnlp)) as phone,  
    QRCNAM as companyName,  
    QRUN as username,  
    qrem1 as email,  
    qrwebu as preference,  
    QSUDAT as createdDate,  
    CASE WHEN QZREFC IS NULL THEN 'ENABLED' ELSE 'DISABLED' END AS status,  
    QSEM2 as createdBy  
    from estesrtgy2.qnp150  
    LEFT JOIN estesrtgy2.qnp230 ON QSUN=QRUN  
    LEFT JOIN ( select QZREFC from estesrtgy2.qnp302 where QZBLTP = 1 and QZREFT = 1) as t   
    ON QZREFC=QSUN  
    WHERE QSEM2=?
  + To get List of Authorized Apps for a user
    - Params:
      * **qnun** : Child Username
      * **qzrefc** (first) : Child Username
      * **qzrefc** (second) : Parent Username (Signed In User's username)
  + select qwmenu, qnun, qntype, qwdsc1   
    from estesrtgy2.qnp200   
    inner join   
    estesrtgy2.qnp220   
    on estesrtgy2.qnp200.qntype=estesrtgy2.qnp220.qwtype   
    where qnun=?   
    and qwmenu not in (select qzapp from estesrtgy2.qnp302 where qzrefc=? and qzbltp=2)   
    and qwmenu not in (select qzapp from estesrtgy2.qnp302 where qzrefc=? and qzbltp=2)   
    and qntype <> 'A12'   
    and qntype<> 'A10'   
    and qntype<> 'A11'   
    and qntype<>'K05'   
    and qntype<>'K01'   
    and qntype <>'F02'   
    and qntype<>'F01'   
    order by qwdsc1
  + To get List of Blocked Apps for a user
    - Params:
      * **qnun** : Child Username
      * **qzrefc** (first) : Child Username
      * **qzrefc** (second) : Parent Username (Signed In User's username)
  + select qwmenu, qntype, qwdsc1  
    from estesrtgy2.qnp200  
    inner join estesrtgy2.qnp220  
    on estesrtgy2.qnp200.qntype=estesrtgy2.qnp220.qwtype  
    where qnun=?  
    and qwmenu in (select qzapp from estesrtgy2.qnp302 where qzrefc=? and qzbltp=2)  
    and qwmenu not in (select qzapp from estesrtgy2.qnp302 where qzrefc=? and qzbltp=2)  
    and qntype <> 'A12'  
    and qntype<> 'A10'  
    and qntype<> 'A11'  
    and qntype<>'K05'  
    and qntype<>'K01'  
    and qntype <>'F02'  
    and qntype<>'F01'  
    order by qwdsc1;
  + To delete all blocked apps for a user
    - Params:
      * **qzrefc** (first) : Child Username
      * **qzrefc** (second) : Parent Username (Signed In User's username)
  + delete from estesrtgy2.qnp302  
    where qzrefc=?  
    AND QZAPP NOT IN (SELECT QZAPP   
     from estesrtgy2.qnp302  
     WHERE QZREFC = ?  
     AND QZBLTP = 2  
     )  
    and qzapp <> 'ADMINSUBS'  
    and qzbltp=2;
  + To delete spcefic apps from User Blocked List
    - Params:
      * **qzrefc** (first) : Child Username
      * **qzrefc** (second) : Parent Username (Signed In User's username)
      * **QZAPP** : App Name
  + delete from estesrtgy2.qnp302  
    where qzrefc=?  
    AND QZAPP NOT IN (SELECT QZAPP   
     from estesrtgy2.qnp302   
     WHERE QZREFC = ?   
     AND QZBLTP = 2  
     )  
    and qzapp <> 'ADMINSUBS'  
    and qzbltp=2  
    AND QZAPP=?;
  + To add an app to user blocked list
    - Params:
      * first : Child Username
      * second : App Name
  + insert into estesrtgy2.qnp302  
    values(?, 1, 2, ?, '', '', CURRENT\_DATE, CURRENT\_TIME, '', '', '', '', '', '', 0, 0);
  + To check if a user is blocked or not
    - Params:
      * **QZREFC** : Username
  + select count(\*) as IS\_BLOCKED from estesrtgy2.qnp302 where QZREFC = ? and QZBLTP = 1 and QZREFT = 1
  + To disable a user
    - Params:
      * first : Child Username
      * Second : Parent Username (Signed In User's Username)
  + insert into estesrtgy2.qnp302  
    values (?, 1, 1, '', '', ?, CURRENT\_DATE, CURRENT\_TIME, '', '', '', '', '', '', 0, 0)
  + To enable a user
    - Params:
      * **qzrefc** : Child Username
  + delete from estesrtgy2.qnp302 where qzrefc = ? and qzbltp = 1 and qzreft = 1
  + To get User Status
    - Params:
      * **QZREFC** : Username
  + select count(\*) as total from estesrtgy2.qnp302 where QZREFC = ? and QZBLTP = 1 and QZREFT = 1
  + To check if a child user is created by Parent User (Signed in User)
    - Params:
      * **createdBY** : Parent Username (Signed In User's Username)
      * **username** : Child Username
  + SELECT count(\*) FROM  
    ( select   
     QRUN as username,  
     QSEM2 as createdBY  
     from estesrtgy2.qnp150  
     LEFT JOIN estesrtgy2.qnp230  
     ON QSUN=QRUN  
     LEFT JOIN (   
     select QZREFC   
     from estesrtgy2.qnp302  
     where QZBLTP = 1   
     and QZREFT = 1  
     ) as t   
     ON QZREFC=QSUN  
    ) AS USER\_INFO  
    WHERE createdBY=?   
    AND username=?