Using Security Questions

Description

This guide outlines a plug-and-play implementation of a **Security Question / Answer** flow within Gigya's RaaS.

Security Question / Answer flow is not fully supported out-of-the-box and requires some additional modification via markup to function properly. Primarily the **UI Builder** does not support mapping to the Gigya Reserved **secretQuestion** and **secretAnswer** fields, so we have to modify the markup directly, and map our input fields to **secretQuestion** and **secretAnswer**. This has to be done on both the **Registration Screen**, as well as the **Registration Completion Screen** within the **RegistrationLogin** Screen-Set. These reserved Gigya fields will automatically be marked as required when they are present on the screen.

Another caveat when using Security Question and Answer is when **passwordReset.requireSecurityCheck** is set to **true** (which is required for using Security Question/Answer), the default **Forgot Password** flow requires manual intervention.

For example, when a user clicks on the **Forgot Password** link, enters their email address, and presses **Submit**, instead of sending a Password Reset email, the following error is returned:

```
{
 "secretQuestion": "Make and model of first car?",
 "errorMessage": "Security Verification Failed",
 "statusCode": 400,
 "errorCode": 400050,
 "statusReason": "Security Verification Failed",
 "callId": "4f6fa0e6c9ff4e998278199b58bfb93f",
 "time": "2016-02-16T09:07:22.593Z",
 "context": "R799361938"
}
```

The solution to this is using the **onAfterSubmit** event handler within our call to accounts.showScreenSet.

We use the following JS code to extract the user's **loginID** and **secretQuestion**.

```
function onAfterSubmitHandler(responseObj){
 if(responseObj.screen === "gigya-forgot-password-screen"){
  if(responseObj.response.errorCode === 400050){
   if(responseObj.response["secretQuestion"] &&
responseObj.response.requestParams["loginID"]){
    console.log(responseObj.response.secretQuestion + "\r\n");
    console.log(responseObj.response.requestParams.loginID + "\r\n");
   }
  }
 }
}
```

Then use accounts.hideScreenSet to close the current dialog before displaying the custom one we created called **gigya-complete-password-reset-screen**.

We load our custom screen and using the **onAfterScreenLoad** event, populate the **loginID** (hidden) and **secretQuestion** fields with the collected data.

If the user has provided the correct **secretAnswer**, the user's password is reset after they **Submit** the form.

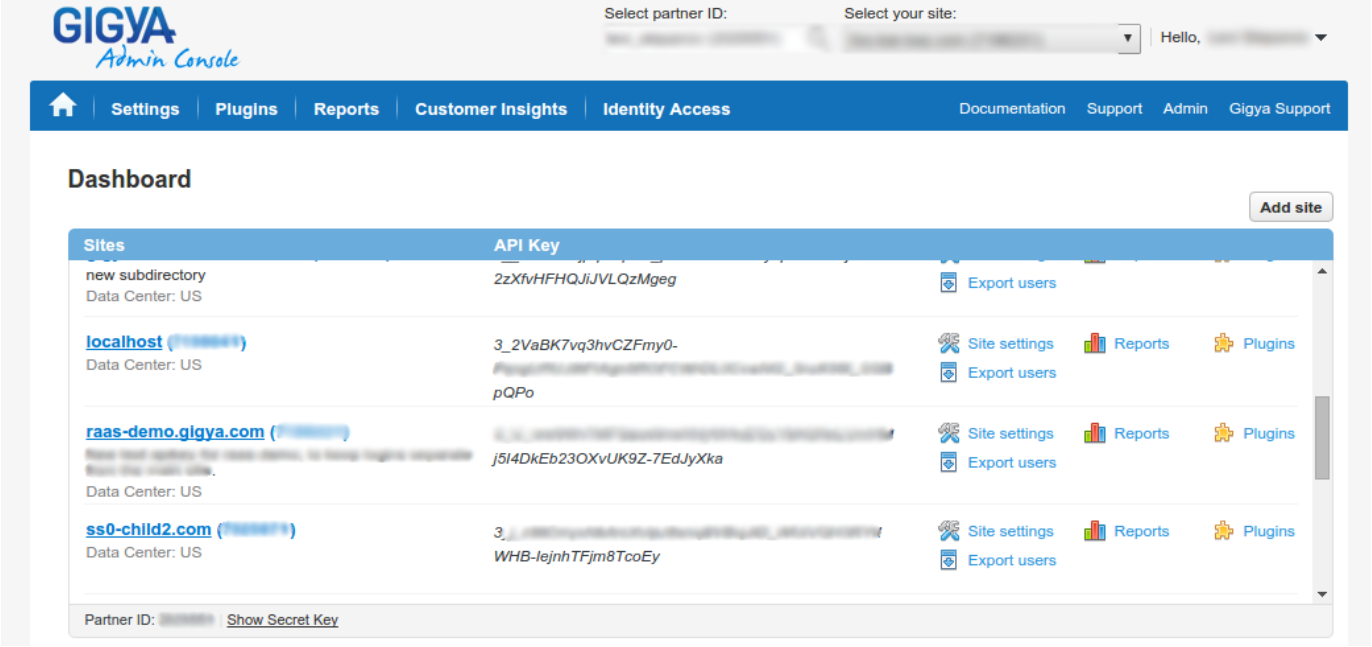
Implementation

To implement this on your site, follow the steps outlined here.

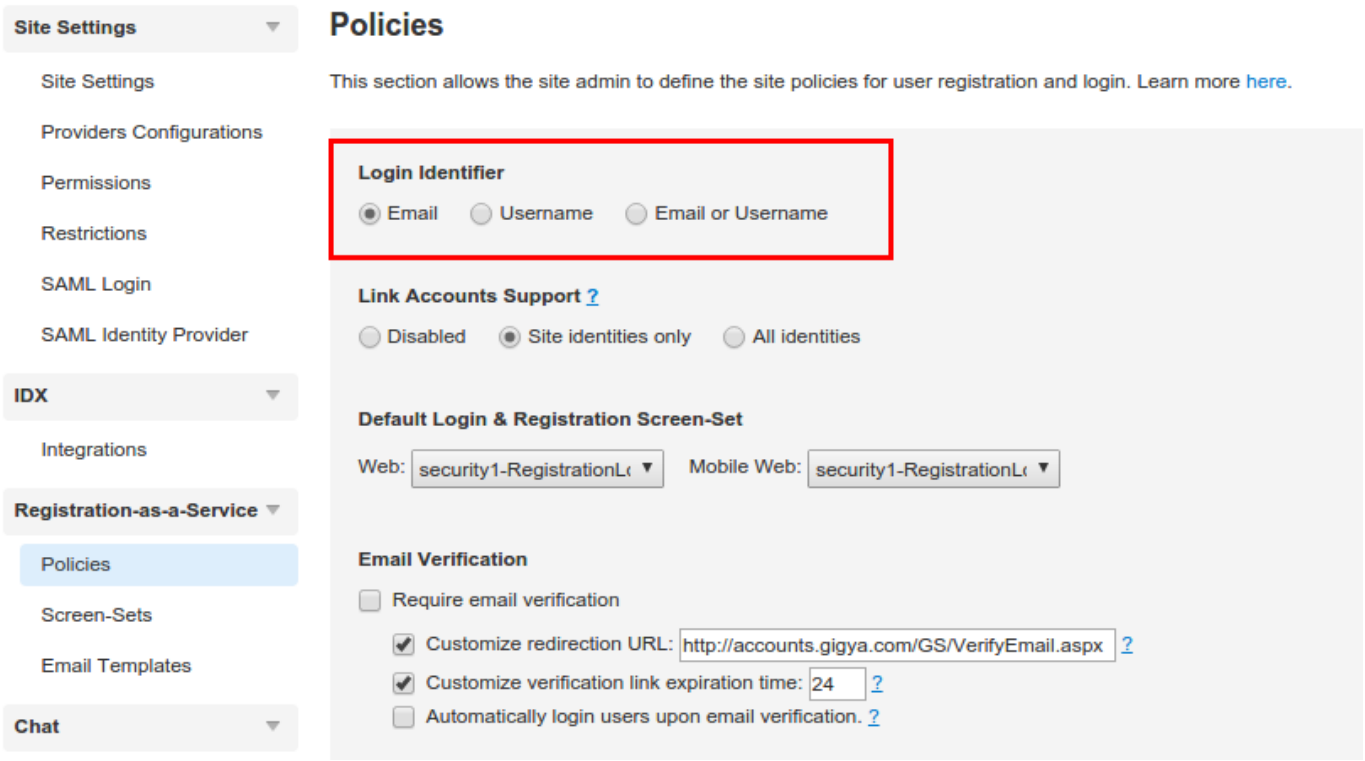
Initial Setup - Stage 1

The first thing that needs to be done is set up your API Key to support the flow.

Do this by opening the **Gigya Console** and navigating to the appropriate **Site** you would like to implement the **Security Question/Answer** flow within.



Then navigate to the **Policies** page of the Dashboard and ensure that you have the Login Identifier set to **Email** (Gigya default).



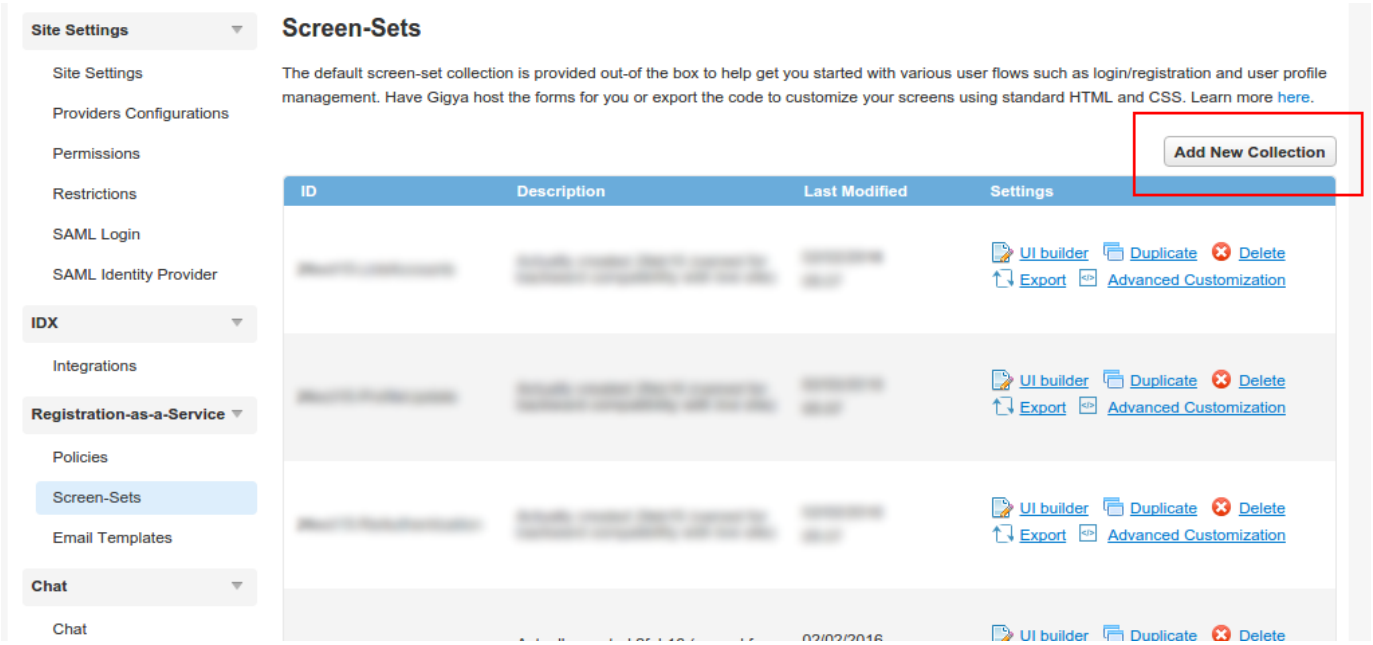
Then use the **accounts.setPolicies** API and set the following parameter:

- **passwordReset.requireSecurityCheck: true**

Creating A Custom Screen-Set Collection

Though the **UI Builder** does not directly support this functionality, you can utilize it to expedite the creation of our custom screen-sets.

Navigate to the **Screen-Sets** tab of the Gigya Dashboard for the API Key from the steps above and add a new screen-set via the **Add New Collection** button.

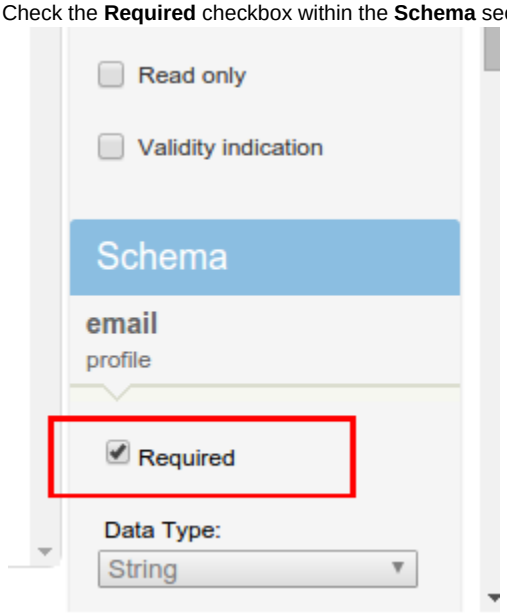


Name this new Collection **security2** and in the description field enter **Security Question/Answer Screens**.

> You can name this screen-set whatever you choose, however, throughout the rest of this document this new screen-set collection will be referred to with the prefix **security2**, containing the following screen-sets:
>
> - **security2-RegistrationLogin**
> - **security2-ReAuthentication**
> - **security2-ProfileUpdate**
> - **security2-LinkAccounts**
>
> If you name this collection differently, be sure to update the downloaded HTML file and change all reference to this collection to the name you have used.

Once created, open the **security2-RegistrationLogin** screen-set in the **UI Builder**.

- Navigate to the **Registration Completion** screen
- Click on the **Email** field to activate its **Properties** window.
- Check the **Required** checkbox within the **Schema** section.



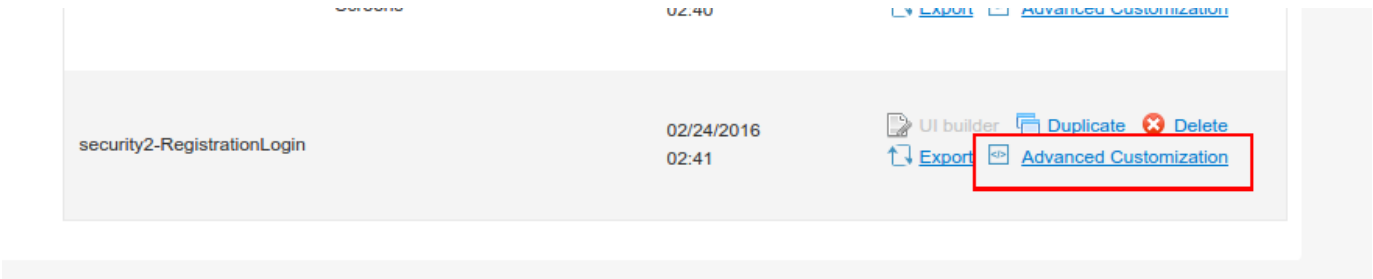
- Press **Save**.

> You will receive a warning notification that this change will effect all screens for this API Key. Read it and select **OK**.

Screen-Set Setup - Stage 2

While still on the **Screen-Sets** tab of the **Gigya Dashboard**, open your favorite text/code editor (gEdit, Notepad++, etc.).

Using the **Advanced Customization** button for the **security2-RegistrationLogin** screen-set:



Download and unzip this file:

security2-RegistrationLogin.html.zip

and copy the contents into the **HTML** tab of the **Security2-RegistrationLogin** screen-set using the **Advanced Customization** tool and press **Save Changes**.

Once you have the security question and answer flow functioning you can further modify this html to suit your company's specific needs, if necessary.

It is recommended to use a static field, such as a **Dropdown**, for the Security Question. This will make it easier to maintain control over the data. Using defined lists will also make it easier for users, as typing in a free-form string different from the previously saved value will cause authentication to fail and possibly make the account unrecoverable.

Implementing the Screen-Sets On Your Site - Stage 3

The code for the working example's **index.html** page is included here.

You can use this as a guide for designing your own login page.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
  <script type="text/javascript" lang="javascript"
src="http://cdn.gigya.com/js/gigya.js?apikey=<Your-API-Key>">
 </script>
 <script type='text/javascript'>
  /**************************************Variable
Declarations**************************************/

  var secretQuestion = "";
  var loginId = "";


/*************************************************************************
***************************/
  /**************************************Login & Logout
Alerts**************************************/
  function onLoginHandler(){
   alert("You are now logged in!");
  }
  function onLogoutHandler() {
   alert('You are now logged out!');
  }
  gigya.accounts.addEventHandlers({onLogin: onLoginHandler, onLogout:
onLogoutHandler});


/*************************************************************************
***************************/
  /**************************************Screen-Set Event
```

```
Handlers************************************/

  function onAfterScreenLoadHandler(responseObj){

   // Check to see if it is the custom password reset screen that was
loaded.
   if(responseObj.currentScreen ===
"gigya-complete-password-reset-screen"){
    // Populate the loginID and secretQuestion fields on the screen.
    document.getElementById("loginIdDiv").value = loginId;
    document.getElementById("secretQuestionDiv").value = secretQuestion;
   }

            document.getElementById("customErrorDivOuter").style.display =
"none";
   secretQuestion = "";
   loginId = "";
  }
  function onAfterSubmitHandler(responseObj){
   if(responseObj.screen === "gigya-forgot-password-screen"){
   /*****************************************************************
    Case I: The regular password-reset screen was loaded.
    In this case, the user is in stage one of password reset,
    so we need to catch the 400050 error and store the loginID
    and secretQuestion that are returned in the response.
   *****************************************************************/

    if(responseObj.response.errorCode === 400050){
     if(responseObj.response["secretQuestion"] &&
responseObj.response.requestParams["loginID"]){
      gigya.accounts.hideScreenSet({screenSet:
"security2-RegistrationLogin"});
      secretQuestion = responseObj.response.secretQuestion;
      loginId = responseObj.response.requestParams.loginID;

      gigya.accounts.showScreenSet({
       screenSet:"security2-RegistrationLogin",
       startScreen:"gigya-complete-password-reset-screen",
       onAfterScreenLoad: onAfterScreenLoadHandler,
       onAfterSubmit: onAfterSubmitHandler
      });
     }
    }
   }

   else if(responseObj.screen === "gigya-complete-password-reset-screen"){
   /*****************************************************************
    Case II: The custom password-reset screen was loaded.
    In this case, the user is in stage two of password reset,
    so we need to catch the 400050 error that means the user
    entered in a wrong secret answer.
   *****************************************************************/
```

```
    if(responseObj.response.errorCode === 400050){
     //These operations trigger the custom error message to prompt the user
to check their secret answer.

document.getElementById("customErrorDivOuter").style.display = "inline";
     document.getElementById("customErrorDivInner").innerHTML = "Security
Validation Failed. Please check your Secret Answer and try again.";
     document.getElementById("customErrorDivOuter").className += "
gigya-error-display-active";
     document.getElementById("customErrorDivInner").className += "
gigya-error-msg-active";
    }
   }

  }

/*************************************************************************
***************************/
 </script>
</head>

<body>
    <span style="font-size: 16px;font-weight: bold;letter-spacing:
1px;float: left;font-family: Arial;">
  <a href="#"
onclick="gigya.accounts.showScreenSet({screenSet:'security2-RegistrationLo
gin', mobileScreenSet:'security2-RegistrationLogin', onAfterSubmit:
onAfterSubmitHandler});">Login</a>  |  
  <a href="#"
onclick="gigya.accounts.showScreenSet({screenSet:'security2-RegistrationLo
gin', mobileScreenSet:'security2-RegistrationLogin',
startScreen:'gigya-register-screen', onAfterSubmit:
onAfterSubmitHandler});">Register</a>    
  </br></br></br>
  <a href="#"
onclick="gigya.accounts.showScreenSet({screenSet:'security2-ProfileUpdate',
mobileScreenSet:security2-ProfileUpdate', onAfterSubmit:
onAfterSubmitHandler});">Edit Your Profile</a>    
  </br></br></br>
  <a href="#"
onclick="gigya.accounts.logout();">Logout</a>    
```

```
  </span>
 </body>
 </html>
```

If you upload your code to Gigya using accounts.setScreenSet it is important to upload both an HTML *and* a CSS file at the same time. Uploading the html only will result in your new screen-sets not being formatted correctly and attempting to upload the css files after uploading the html will cause the html to be deleted.

If making changes to both HTML and CSS from within the UI Builder Advanced Customization tool it is important to save and re-open the screen-set between each operation of changing the HTML or the CSS. Switching between tabs while the **Advanced Customization** tool is open will result in the previously viewed tab being reset to it's original content.

Working Example

Download the source files

Index.html.zip

security2-RegistrationLogin.html.zip

security2-RegistrationLogin.css.zip

If only using the **UI Builder Advanced Customization** tool you do not need the CSS file unless you would like to change the default Gigya CSS.