VERSION 1.2

JULY 1, 2019

swipeclock

WORKFORCE MANAGEMENT

LOFT APPLICATION FRAMEWORK

DEVELOPER GUIDE

TABLE OF CONTENTS

OVERVIEW

The HUB App Framework, named LOFT, is an extensible plug-in framework allowing third party developers to create and integrate a rich set of functionality for HUB users. The framework is modeled off of what Apple and Google have done with their iTunes and Play stores as well as what Facebook has done with third party apps.

WHY CREATE AN APP?

HUB is the front-end user experience offered by independent payroll providers across the country. Employers and employees access HUB to conduct basic payroll tasks.

Employers utilize HUB to view payroll reports, administer time and attendance, approve timesheets, review PTO requests, add new hires, conduct base employee maintenance and communicate to employees through message boards and document libraries. Employees utilize HUB to view pay stubs and W2s/1099s, clock in and out, request PTO and access important company information.

Apps will allow any extension to HUB that an employer or employee needs. Things such as expense management, benefit enrollment and management, retirement accounts, pay card account access and company chat are all examples of apps that will help round out a unique payroll experience for employers. HUB’s App Framework now enables these extensions to be a reality.

WHO CAN CREATE AN APP?

Any third party with basic web development knowledge can create an app to be available within HUB. An interested third party must register with SwipeClock as an app developer. Only authorized app developers will have access to a sandbox account to create new apps and update existing apps for approval. The approval process is what will determine authorization of an app by SwipeClock to be promoted from the sandbox environment to production.

REGISTERING FOR A DEVELOPER ACCOUNT

All interested parties in becoming an app developer must register for a developer account by requesting/submitting an application form through developer@swipeclock.com. SwipeClock reserves the right at its sole discretion to approve or reject all submissions. Upon approval, SwipeClock will provide access to a sandbox environment utilized for creation of your app.

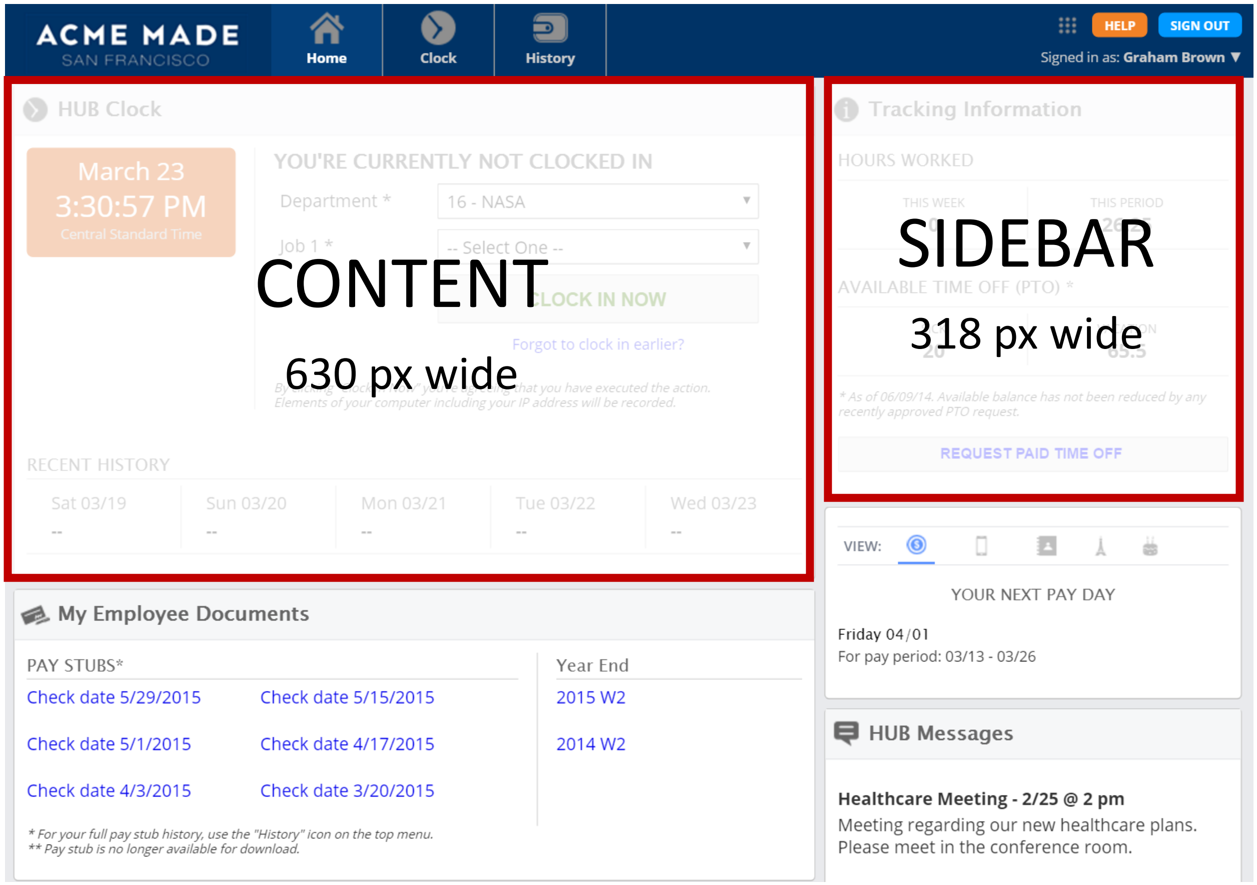
COMPOSITION OF AN APP

An app in the HUB App Framework is comprised of a website, housed under SSL, hosted on an external server. SwipeClock will not host apps, as it's the third-party developer's responsibility. All apps will be rendered as embedded iFrames within HUB allowing for the de-coupling of HUB resources from the app's resources.

APP SIZES

Apps must fit into one of the two categories/sizes below:

- Content: maximum of 630 pixels wide.
- Sidebar: maximum of 318 pixels wide.



Friday 04/01
For pay period: 03/13 - 03/26

Check date 5/29/2015 | Check date 5/15/2015 | 2015 W2
Check date 5/1/2015 | Check date 4/17/2015 | 2014 W2
Check date 4/3/2015 | Check date 3/20/2015

* For your full pay stub history, use the "History" icon on the top menu.
** Pay stub is no longer available for download.

Healthcare Meeting - 2/25 @ 2 pm
Meeting regarding our new healthcare plans. Please meet in the conference room.

Each app can be variable in height, but should not attempt to take up the majority of HUB's screen real estate. SwipeClock's app approval process will determine if the app's size is appropriate.

CREATING AN APP

An app is a publicly accessible URL under SSL. An app must reference HUB's JavaScript SDK and may optionally take advantage of HUB's REST APIs.

JAVASCRIPT SDK

The JavaScript SDK provides the first integration point of an app within HUB. It provides initial app verification and returns information on the current user.

The code below shows how to integrate the JavaScript SDK using JQuery within an app. This is a very simple app which just says "Hey there" to the current HUB user.

```
<html>
<head>
    <title>Hi User App</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
    <script type=”text/javascript”>
        $(function () {
            $.ajaxSetup({ cache: true });
            $.getScript('//[pathToJavaScriptSdk]/hub-sdk.js', function () {
                HUB.init({ widgetKey: 'com.myshugo.myappname', version: '1.0.0.0',
height: '200px', width: '318px'}, function (hubApp) {
```

```
                    // Code to do the work of the app
                    $('body').text('Hey there ' + hubApp.user.FirstName + ' ' +
hubApp.user.LastName);
                });
            });
        });
    </script>
</head>
<body>
</body>
</html>
```

Breaking down the app, the following occurs:

1. Retrieval of the HUB App Framework SDK occurs using the JQuery $.getScript method.
2. The App Framework SDK is initialized in the *HUB.init* call. The parameters include the current app's options (assigned widget key, current app version number, requested height and width for display) and the callback function to execute on successful authorization of the app. The callback function in this case just adds the text "Hey there" to the current user in the app's body tag.

The URL for the path to the JavaScript SDK will be provided to developers upon approval of their developer account submission.

HUB.INIT

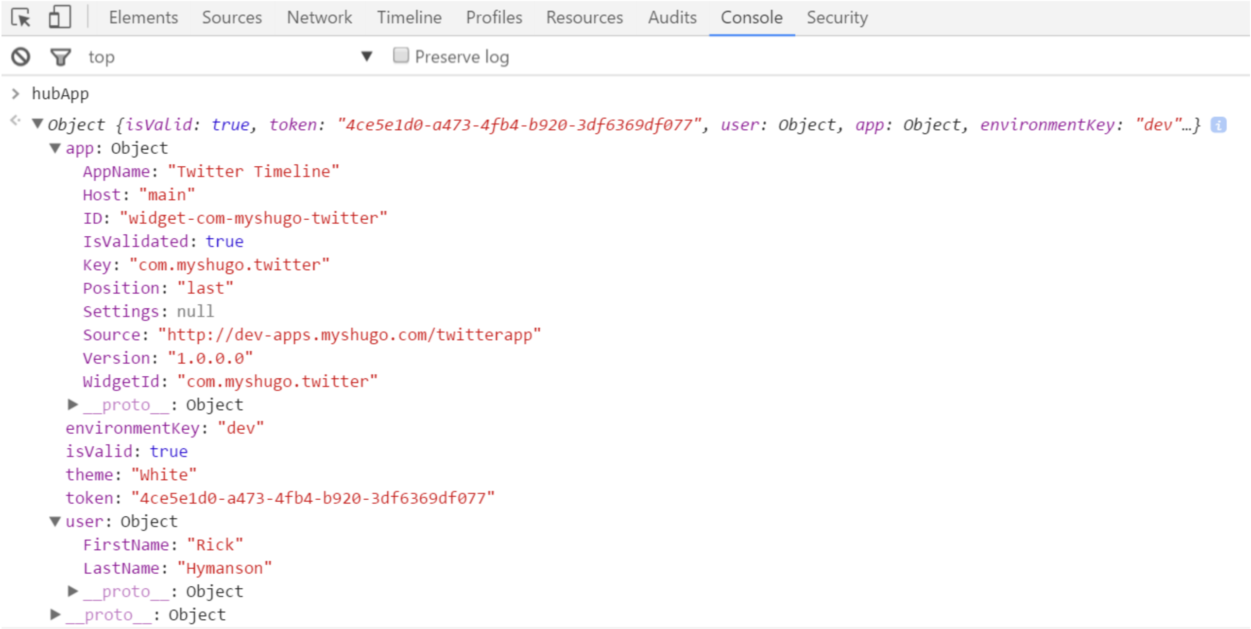
The init method is used to initialize and setup the SDK and must be called each time the app is rendered (i.e. each time the app's URL is called).

```
HUB.init({ widgetKey: 'com.myshugo.myappname', version: '1.0.0.0', height: '200px',
width: '318px'}, function (hubApp) {});
```

Name	Type	Description
params	object	A collection of initialization parameters that control the setup of the SDK. Required properties include: • widgetKey (string): Your application widget key assigned by SwipeClock. • version (string): The version number of your app. This must match the exact version number of your app as shared with SwipeClock. • height (string): requested height in pixels of the application in HUB. • width (string): requested width in pixels of the application in HUB.

Name	Type	Description
callback function	Function	A function for the SDK to execute upon successful initialization of the app. The callback function will receive one parameter: • hubApp (object): Instance of a HUB application which includes the following: ○ app (object): information about your application configured within HUB. ○ environmentKey (string): identification of the deployment environment you're currently in. Valid values include: dev, qa, prod. ○ theme (string): chosen color scheme applied to the user's HUB interface. Examples include: Neutral, White, Red, Blue, Yellow, Green, Silver, etc… ○ isValid (Boolean): true/false indicator on the validity of your init request. ○ token (string): current rendered page token that will be used if making additional calls to SwipeClock REST APIs. ○ user (object): high level information(first and last name) about the current user. The callback function will execute the main code of your app. If initialization of your application fails within the SDK, the callback function will never be executed.

The following shows a console view of the hubApp returned in the callback function:

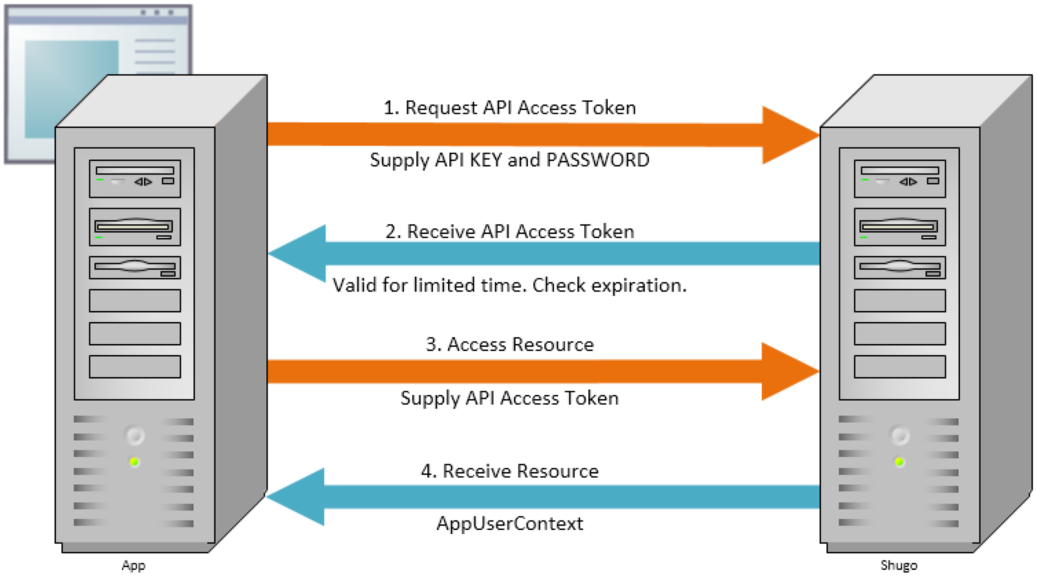


```
> hubApp
< ▼Object {isValid: true, token: "4ce5e1d0-a473-4fb4-b920-3df6369df077", user: Object, app: Object, environmentKey: "dev"…}
   ▼app: Object
      AppName: "Twitter Timeline"
      Host: "main"
      ID: "widget-com-myshugo-twitter"
      IsValidated: true
      Key: "com.myshugo.twitter"
      Position: "last"
      Settings: null
      Source: "http://dev-apps.myshugo.com/twitterapp"
      Version: "1.0.0.0"
      WidgetId: "com.myshugo.twitter"
     ▶__proto__: Object
    environmentKey: "dev"
    isValid: true
    theme: "White"
    token: "4ce5e1d0-a473-4fb4-b920-3df6369df077"
   ▼user: Object
      FirstName: "Rick"
      LastName: "Hymanson"
     ▶__proto__: Object
   ▶__proto__: Object
```

REST API

The HUB App Framework REST API allows you to query data from HUB in context of the current User. The API is served over HTTPS and responses are available in JSON or XML format, depending on the request's Content-Type header. URLs for the REST API will be provided to parties approved for a developer account by SwipeClock.

API WORKFLOW


1. Request API Access Token
Supply API KEY and PASSWORD
2. Receive API Access Token
Valid for limited time. Check expiration.
3. Access Resource
Supply API Access Token
4. Receive Resource
AppUserContext
App
Shugo

AUTHENTICATION

All REST API requests must contain a valid API session token. This token is generated by supplying an API Key and Password to the Authentication endpoint (see POST /tokens below). All authorized third-party developers will be assigned an API Key and Password by SwipeClock. Requests to the Authentication API are protected with HTTP Authentication.

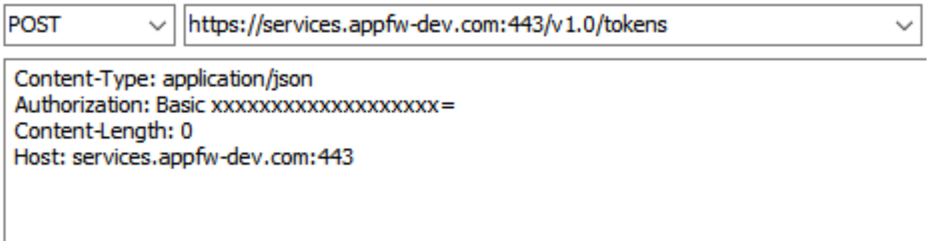
GENERATE TOKEN ENDPOINT

REQUEST

URL:	/tokens
Description:	Generates a new API token for the third party developer. Tokens expire after a period of time.
HTTP Method:	Post
HTTP Headers:	Content-Type | application/json

		or application/xml
	Authorization	Basic APIKEY:APIPASSWORD *Note that APIKEY:APIPASSWORD must be Base64 encoded.*

A sample request is shown below.



```
https://services.appfw-dev.com:443/v1.0/tokens
```

```
Content-Type: application/json
Authorization: Basic xxxxxxxxxxxxxxxxxx=
Content-Length: 0
Host: services.appfw-dev.com:443
```

RESPONSE

The response body will be formatted in with JSON or XML format based on the request's Content-Type HTTP Header.

HTTP Status Code:	200: successful authentication 400: Invalid header or credentials supplied 401: Request was blocked	
Body:	Token	String A valid API token to use on subsequent REST API calls.
	ExpiresUTC	String A token's expiration date in UTC format.

A sample response is depicted below.

```
{
  "Token": xxx-xxxx-xxxxxx-xxxxx,
  "ExpiresUTC": "2-11-2016 22:40:00",
}
```

USER CONTEXT RETRIEVAL

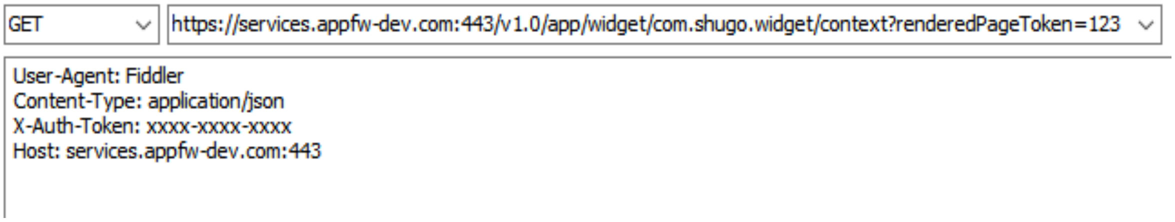
User context retrieval provides the developer detailed information on the current HUB user. It provides unique identifiers and attributes of the payroll provider (firm), employer (company) and employee (user) currently using HUB.

GET APP WIDGET USER CONTEXT

REQUEST

URL:	/app/widget/{widgetKey}/context?renderedPageToken={renderedPageToken}	
Description:	Retrieves the user context information of the current HUB user who has had the app served to them on a HUB page.	
HTTP Method:	Get	
HTTP Headers:	Content-Type	application/json or application/xml
	X-Auth-Token	A valid authentication token obtained from the Authentication request.
Parameters:	widgetKey	The identifier of your app widget provided to SwipeClock when creating your app.
	renderedPageToken	Token generated by HUB and returned to your app using the JavaScript SDK's HUB.init method.

A sample request is shown below.



```
GET
https://services.appfw-dev.com:443/v1.0/app/widget/com.shugo.widget/context?renderedPageToken=123

User-Agent: Fiddler
Content-Type: application/json
X-Auth-Token: xxxx-xxxx-xxxx
Host: services.appfw-dev.com:443
```

RESPONSE

The response body will be formatted in with JSON or XML format based on the request's Content-Type HTTP Header.

<table>
<tr><td><b>HTTP Status Code:</b></td><td>200: successful authentication
400: Invalid header or parameters supplied
401: Invalid or expired page token supplied</td></tr>
<tr><td><b>Body:</b></td><td>
<table>
<tr><td>Theme</td><td>String
Color scheme currently applied to HUB’s user interface (allows the developer to apply their own styles to match the current color scheme).</td></tr>
<tr><td>InterfaceType</td><td>String
Identifies the type of interface currently being used to access HUB: Valid values include: Web and Mobile.</td></tr>
<tr><td>AppSettings</td><td>Object
A list of any configured application settings.
<table>
<tr><td><b>AppSettingName</b></td><td>String</td><td>The name of the setting.</td></tr>
<tr><td><b>AppSettingValue</b></td><td>String</td><td>The configured value of the setting.</td></tr>
</table></td></tr>
<tr><td>Company</td><td>Object
Identification of the employer that the current user has accessed in HUB.
<table>
<tr><td><b>Code</b></td><td>String</td><td>Unique code (per payroll provider) assigned to this employer within the payroll system.</td></tr>
<tr><td><b>Name</b></td><td>String</td><td>Legal name of the employer.</td></tr>
<tr><td><b>Address</b></td><td>Object</td><td>Legal address of the employer.</td></tr>
</table></td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td></td>
<td>Firm</td>
<td>Object
Identifies the payroll provider that the current user has accessed in HUB.
<table>
<tr><td><b>Code</b></td><td>String</td><td>Unique code assigned to this payroll provider by SwipeClock.</td></tr>
<tr><td><b>Name</b></td><td>String</td><td>Name of the payroll provider.</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>User</td>
<td>Object
Identifies the current user within HUB. Note that the current user could be a business owner or employee or third party (ie. accountant) who has access to the current employer (company).
<table>
<tr><td><b>Code</b></td><td>String
(optional)</td><td>Not used.</td></tr>
<tr><td><b>Name</b></td><td>String</td><td>Full name of the user.</td></tr>
<tr><td><b>Employee Number</b></td><td>String
(optional)</td><td>Unique identifier assigned to this employee within the context of their employer (company) from the payroll system.</td></tr>
<tr><td><b>First Name</b></td><td>String</td><td>First name of the user.</td></tr>
<tr><td><b>Gender</b></td><td>String
(optional)</td><td>User’s gender. Valid values include:
• Male
• Female
• Not Reported</td></tr>
<tr><td><b>Home Address Zip Code</b></td><td>String
(optional)</td><td>Five digit zip code of the employee’s home address.</td></tr>
<tr><td><b>Role List</b></td><td>String Array</td><td>User’s assigned roles within HUB. Value values include:
• Administrator
• Firm Representative
• Employee
• Manager</td></tr>
</table>
</td>
</tr>
</table>

A sample response is depicted below.

```
{
      "Theme" : "Blue”,
      "InterfaceType" : "Web",
      "AppSettings" : [{
                   "AppSettingName" : "Setting",
                   "AppSettingValue" : "Value"
             }
      ],
      "Company" : {
             "Code" : "1234",
             "Name" : "Test, Inc.",
             "Address" : {
                   "Address1" : "123 Main St.",
                   "Address2" : null,
                   "City" : "Anywhere",
                   "StateCode" : "NJ",
                   "ZipCode" : "01010",
                   "ZipPlus4" : null
             }
      },
      "Firm" : {
             "Code" : "9",
             "Name" : "Contoso Payroll"
      },
      "User" : {
             "Code" : null,
             "Name" : "John Doe",
             "Email" : "john.doe@myshugo.com",
             "EmployeeNumber" : "1111",
             "FirstName" : "John",
             "Gender" : "Not Reported",
             "HomeAddressZipCode" : null,
             "LastName" : "Doe",
             "RoleList" : ["Employee", "Administrator"]
      }
}
```

APP STYLING

An app can create its own sense of style but it's recommended to adhere to the existing style rules currently in place within HUB itself. Below are styles currently in use in HUB, which if used would ensure consistency in your application with HUB.

```
html, body {
    margin: 0;
    font-size: 14px;
    font-family: "Lucida Grande" , "Lucida Sans Unicode" , helvetica, arial, verdana,
sans-serif;
}

h1, h2, h3, h4, h5 {
    margin-top: 0;
    margin-bottom: 0;
    line-height: 150%;
    color: #666666;
}

div.app-host {
    background-color: #fff;
    border: solid 1px #ccc;
    border-bottom-color: #c0c0c0;
    -moz-border-radius: 4px;
    -webkit-border-radius: 4px;
    border-radius: 4px;
    -khtml-border-radius: 4px;
    margin-bottom: 7px;
    padding-bottom: 10px;
}
```

```
div.std-layout {
    padding: 8px;
}

div.app-host > h3 {
    background-color: #f7f7f7;
    border-bottom: 1px solid #f2f2f2;
    text-align: left;
    padding: 8px 10px 8px 35px;
    background-position: 5px;
    background-repeat: no-repeat;
}
```

The HTML below shows usage of these styles:

```
    <div class="app-host">
        <h3 class="twitter">Latest News and Tweets</h3>
        <div class="std-layout"></div>
    </div>
```

UNDERSTANDING THEMES

HUB supports themes which allows preferred color schemes to be applied to the user's HUB interface. The applied theme is returned in both the JavaScript SDK and REST API. The below styling guide provides more detailed CSS information based on the selected color scheme and uses some of the CSS classes above in the App Styling section.

Theme Name	Details	

Neutral	Primary Colors	background-color: #333333;
		color: #FFF;

	div.app-host > h3	background-color: #F7F7F7;
		border-bottom: 1px solid #F2F2F2;

Blue	Primary Colors	background-color: #003366;
		color: #FFF;

	div.app-host > h3	background-color: #F6F7F8;
		border-bottom: 1px solid #E9EAED;

<table>
<tr><th>Theme Name</th><th colspan="2">Details</th></tr>
<tr><td rowspan="2">Red</td><td>Primary Colors</td><td>background-color: #AA2B2B;
color: #FFF;</td></tr>
<tr><td>div.app-host > h3</td><td>background-color: #F6F7F8;
border-bottom: 1px solid #E9EAED;</td></tr>
<tr><td rowspan="2">Silver</td><td>Primary Colors</td><td>background-color: #CCCCCC;
color: #444444;</td></tr>
<tr><td>div.app-host > h3</td><td>background-color: #DDD;
border-bottom: 1px solid #F2F2F2;</td></tr>
<tr><td rowspan="2">White</td><td>Primary Colors</td><td>background-color: #FFF;
color: #333;</td></tr>
<tr><td>div.app-host > h3</td><td>background-color: #F9F9F9;
border-bottom: 1px solid #DDD;</td></tr>
<tr><td rowspan="2">Green</td><td>Primary Colors</td><td>background-color: #006600;
color: #FFF;</td></tr>
<tr><td>div.app-host > h3</td><td>background-color: #61B329;
border-bottom: 1px solid #E9EAED;</td></tr>
<tr><td rowspan="2">Yellow</td><td>Primary Colors</td><td>background-color: #FFFF46;
color: #333;</td></tr>
<tr><td>div.app-host > h3</td><td>background-color: #FFFF46;
border-bottom: 1px solid #E9EAED;</td></tr>
</table>

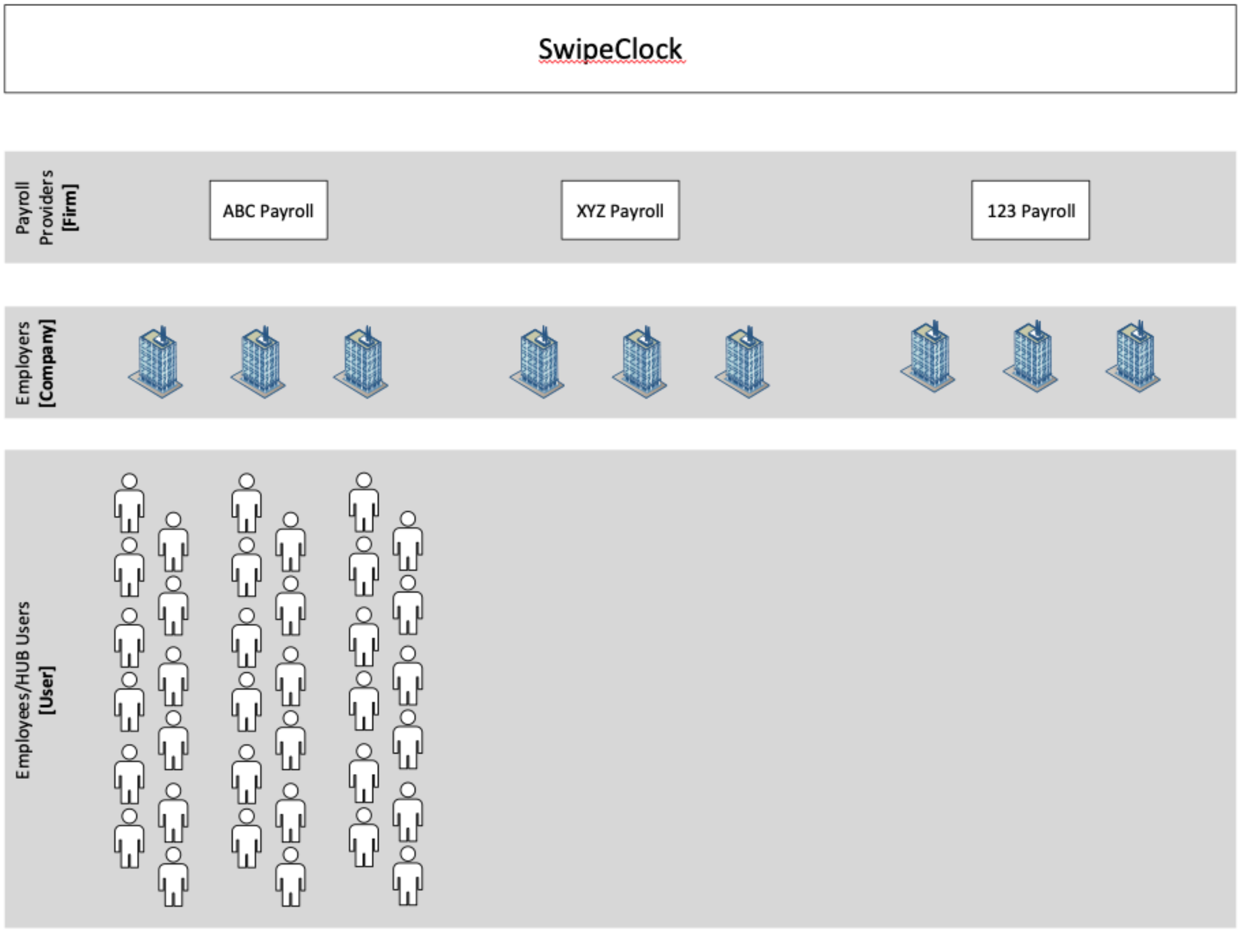
Theme Name	Details	

Purple	Primary Colors	background-color: #443266;
		color: #FFF;

	div.app-host > h3	background-color: #F1F0FF;
		border-bottom: 1px solid #E9EAED;

APPENDIX

HUB ENTITIES


SwipeClock
Payroll Providers [Firm]
ABC Payroll
XYZ Payroll
123 Payroll
Employers [Company]
Employees/HUB Users [User]