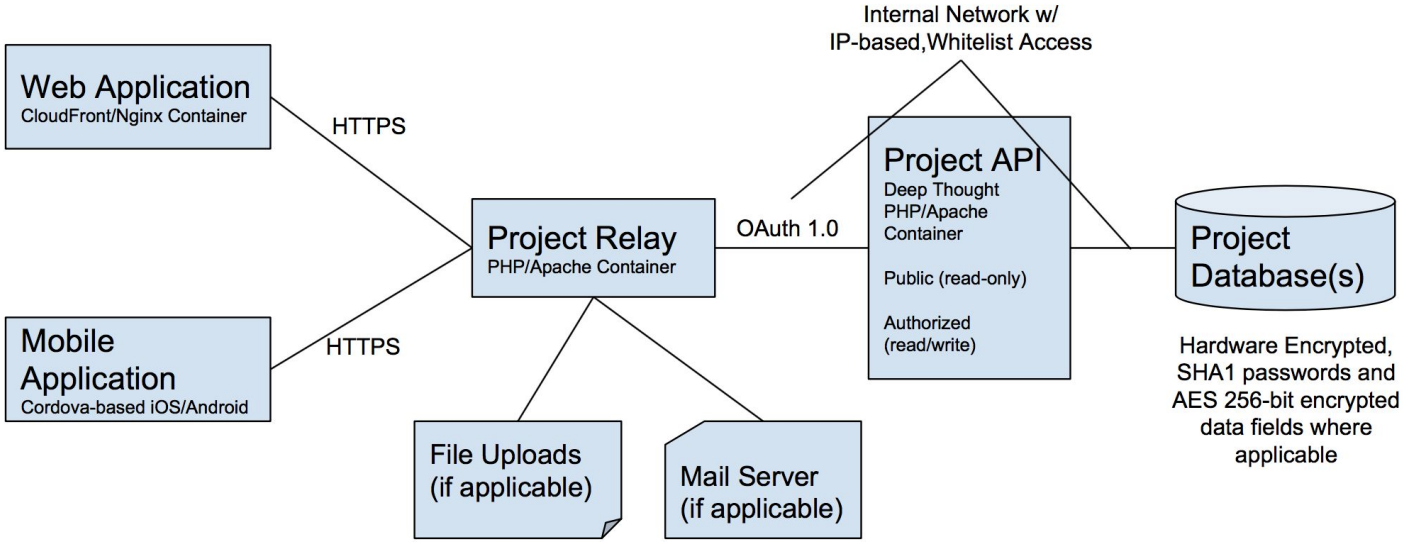
Confide

Single-Page Application w/ Secure API Communication Using DTRelay

DTRelay Sample Architecture



Amazon Web Services EC2s + RDS (Postgres/MySQL) in Docker Swarm Configuration

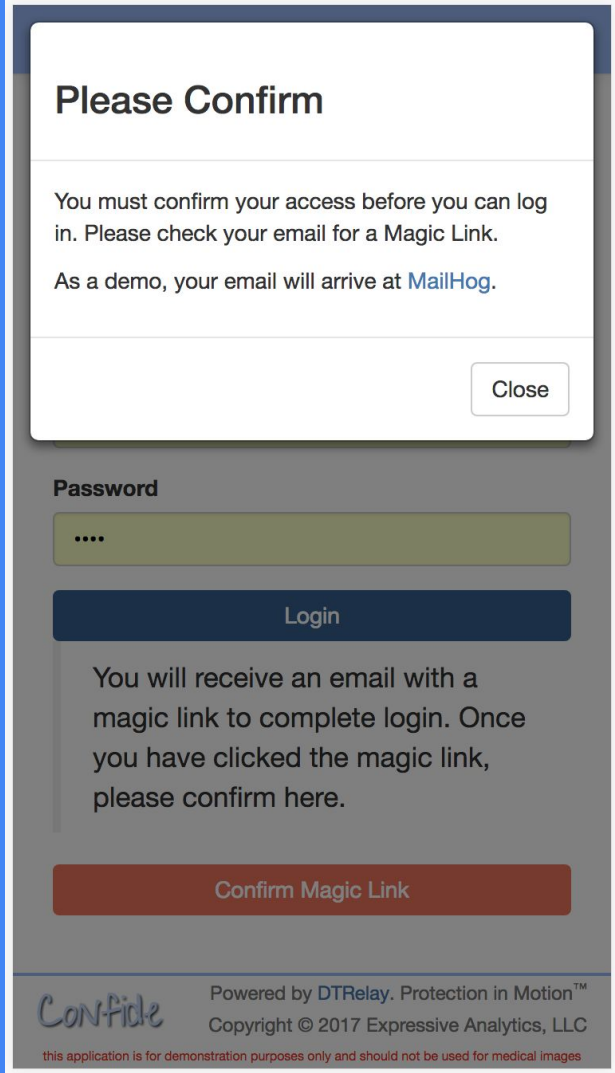
API Configuration

Connect to Public or Secure/OAUTH-based APIs

```
{
  "api": {
    "url": "http://api.local/api/",
    "delegates": [
      "DTProviderDelegate",
      "MFADelegate"
    ],
    "consumer_key": "[redacted]",
    "consumer_secret": "[redacted]"
  },
  "api-oauth": {
    "url": "http://api.local/api-oauth/",
    "delegates": [
      "DTProviderDelegate",
      "DTOAuth1Delegate",
      "MediaDelegate"
    ],
    "consumer_key": "[redacted]",
    "consumer_secret": "[redacted]",
    "request_token_url": "http://api.local/api-oauth/user/request_token",
    "authorize_url": "login",
    "callback_url": "http://localhost/api-oauth/",
    "access_token_url": "http://api.local/api-oauth/user/access_token"
  },
  "aws-s3":{
    "url": "us-west-2",
    "consumer_key": "[redacted]",
    "consumer_secret": "[redacted]",
    "bucket": "dt-development",
    "path": "sandbox"
  },
  "facebook":{
    "url":"https://graph.facebook.com/v2.7/",
    "delegates": ["DTOAuth2Delegate"],
    "consumer_key": "[redacted]",
    "consumer_secret": "[redacted]",
    "authorize_url": "https://www.facebook.com/dialog/oauth",
    "redirect_uri": "http://localhost:8887/facebook/",
    "access_token_url": "https://graph.facebook.com/oauth/access_token",
    "response_type": "code",
    "scope": "public_profile"
  },
  "twitter":{
    "url": "https://api.twitter.com/1.1/",
    "delegates": ["DTOAuth1Delegate"],
    "consumer_key": "[redacted]",
    "consumer_secret": "[redacted]",
    "request_token_url": "https://api.twitter.com/oauth/request_token",
    "authorize_url": "https://api.twitter.com/oauth/authorize",
    "callback_url": "http://localhost:8887/twitter/",
    "access_token_url": "https://api.twitter.com/oauth/access_token",
    "handle": "@photrist"
  },
  "google":{
    "url": "https://www.googleapis.com/oauth2/v2/",
    "delegates": ["DTOAuth2Delegate"],
```

Multi-Factor Authentication

MFADelegate



MFADelegate

- Simple relay delegate to trigger multi-factor authentication (MFA)
- During standard authentication (user login), it handles MFA:
  - We have configured the delegate to require 2 factors of authentication
  - Step 1: send an *authorize_factor* request for the user's ID
  - Step 2: send a magic link via user's email address
- We can easily support any authorizer that contains a valid identifier

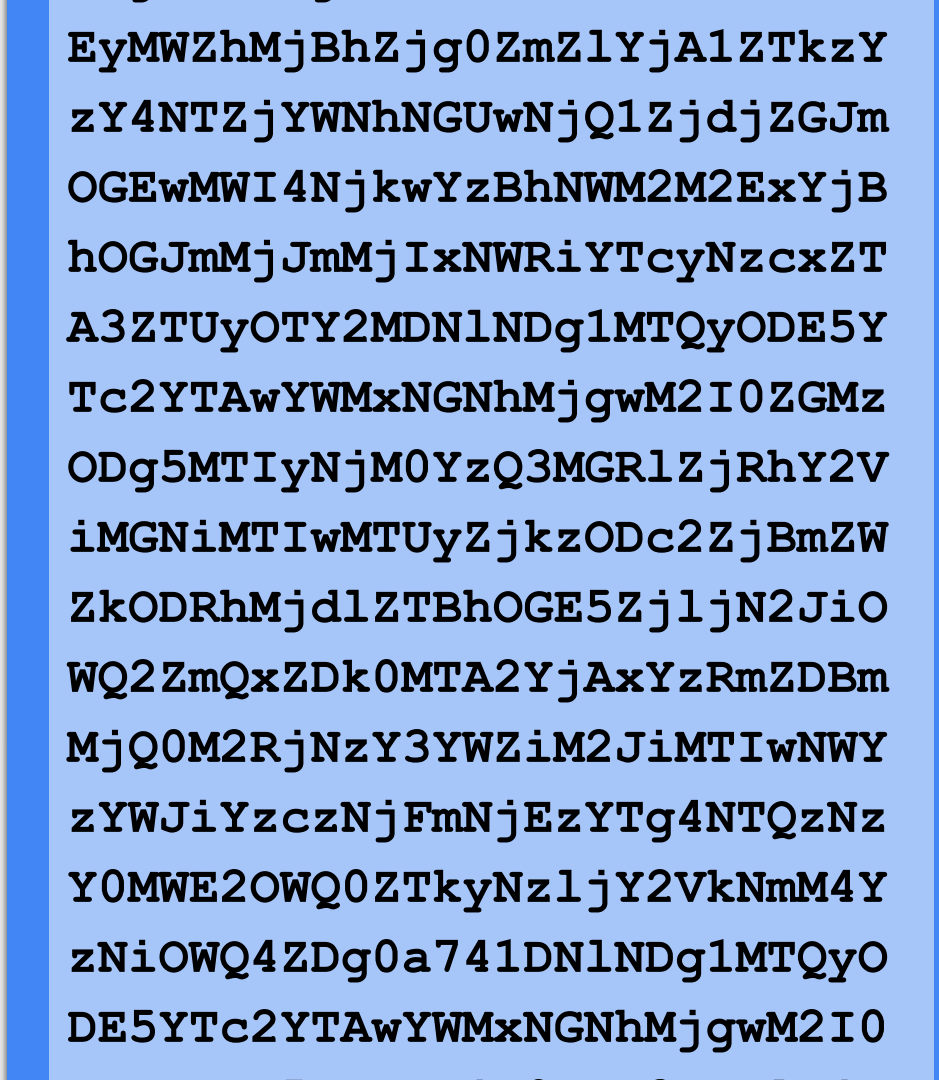
```php
public function postprocess(&$rsp,&$obj){
  global $config;
  if($this->cmr->endpoint=="authentication" && $this->cmr->action=="authenticate"){
    $tok = $obj["obj"];
    if($tok["user_id"]){ // we have successfully logged in
      $tok_str = $tok["token"];

      // STEP 1: authorize "id" factor
      // Since there is a race condition for request_tokens, we need to save
      // the correct MFA token for future authorizers.
      $tgt = DTAPI::consumer("api-oauth");
      $this->cmr->keystore->set($tgt->api,"mfa_token",$tok_str);
      $this->cmr->action("authorize_factor")->post(array(
        "key"=>"id","val"=>$tok["user_id"],"oauth_token"=>$tok_str
      ));

      // STEP 2: send magic link via email
      $mailer = new DTPHPMailer(null,null);
      $to = new DTEmailRecipientList();
      $to->setAddresses(array("email"=>$this->email));
      $subject = "Please Confirm Login Attempt";
      $magic_link = $config["relay_url"]."api/authentication?act=authorize_factor&oauth_token={$tok_str}&key=magic_link";
      ob_start(); //capture the following as HTML
      require_once(dirname(__FILE__)."/../email/magic_link.php");
      $html = ob_get_contents();
      ob_end_clean();
      $mailer->sendHTMLEmail($to,$subject,$html);
    }
  }
}
```

Secure Image Encryption

DTRelay's Secret + MediaDelegate



Protection in Motion ™

- During initialization, the relay provides the client application with a secret
- We can use this secret value to encrypt our images during transfer
- The relay decrypts the image and processes it as a standard upload
- This same technique is used to encrypt any “protected parameters” so that they are tamper-proof and undecipherable in-transit (even without SSL)

Client-side Encryption (AES)

Using DTRelay's secret value as the encryption key, and the request's unique nonce value as the initialization vector, we use AES to encrypt the images during transfer.

```
    Relay.get("SecureMedia",params,function(response){
      $ctrl.item = response.obj // populate the object
      Relay.getAPIs().then(function(apis){
        var api = apis["SecureMedia"]
        var params = {
          "act":"download",
          "id":$ctrl.item.id,
          "api":api.api,
          "end":api.end
        }
        if(typeof($stateParams.review)!=="undefined")
          params.review = true
        var params = Relay.insertTokens(params)
        $http.get(api["url"],{"params":params})
          .then(function(response,headers){
            var sec = localStorage.getItem("RelaySecret")
            var data = CryptoJS.AES.decrypt({
              ciphertext:CryptoJS.enc.Hex.parse(response.data),
              salt:""
            },CryptoJS.enc.Utf8.parse(sec),{
              iv:CryptoJS.enc.Utf8.parse(params.dtnnc)
            })
            if(data.length>0)
            $ctrl.item.data64 = "
            "+data.toString(CryptoJS.enc.Utf8)
          })
      })
    })
```