HAPI iOS SDK Tutorial

Description

The iOS app provides a sample iOS application using HAPI to authenticate and access on-premise Active Directory files and folders from an iOS device. Files can be downloaded to the device or local device files can be upload to shares published in Active Directory.

Tutorial

Introduction

This sample iOS app is written using Swift in XCode and demonstrates how to use HAPI to login, navigate files and folder published in Active Directory, download a file to the device, and upload a local device file into Active Directory. The app was written in Swift using XCode 6.4, requires framework in iOS 5.0 and later. By default, the app makes calls to the HAPI Gatekeeper in our sandbox environment, which is supported by a HAPI Gateway and sandbox Active Directory instance. FullArmor can supply a temporary sandbox Active Directory account, complete with sample data, upon request.

The HAPI interface is defined on our API Documentation pages at https://sandbox.fullarmorhapi.com/route/swagger and https://sandbox.fullarmorhapi.com/swagger. These are Gateway and Gatekeeper calls respectively. All HAPI calls are directed to the Gatekeeper. Urls with /route/hapi will be routed to the proper Gateway. Urls with /route/agent/<id> will be routed to the proper HAPI Agent. Urls with /api access the interface presented by the Gatekeeper.

App Operations

Login

HAPI Calls: Login (/route/hapi/Login) Post

HAPI login takes Active Directory credentials and creates a token required for all other HAPI calls. In the sandbox environment, the token only expire with 15 minutes of idle time. This sample app, will show a login page if the token does not exist or has expired.

Here is code from the sample app performing a login.

```
func  PromptForCreds()
  {
      let  username =  KeychainHelper.loadString("userName")
      if  username !=  nil
     {
         let  password =  KeychainHelper.loadString("password")
         if  password !=  nil
        {
            let  requestUrl =  KeychainHelper.loadString("serverUrl")
            if  requestUrl !=  nil
           {
               if  (HAPIClient.sharedInstance.Login(requestUrl!, userName: username, password:
password))
```

```
            {
              self.serverUrl  = requestUrl
             self.LoadADShares()
              return;
            }
          }
        }
      }
       let  alertController =  UIAlertController(title:  "Default Style", message:  "A standard alert.",
                     preferredStyle: .Alert);
      let  loginAction =  UIAlertAction(title:  "Login", style: .Default)
      {
          (_)  in
           let  serverTextField = alertController.textFields![0]  as!  UITextField
           let  usernameTextField = alertController.textFields![1]  as!  UITextField
           let  passwordTextField = alertController.textFields![2]  as!  UITextField

           if(  HAPIClient.sharedInstance.Login( serverTextField.text, userName: usernameTextField.text,
                     password: passwordTextField.text) )
          {
             self.serverUrl  = serverTextField.text
             KeychainHelper.saveString("serverUrl", data:serverTextField.text)
             KeychainHelper.saveString("userName", data: usernameTextField.text)
             KeychainHelper.saveString("password", data: passwordTextField.text)

             return
          }
           else
          {
             self.PromptForCreds();  // Login failed, prompt again
          }
      }

      alertController.addTextFieldWithConfigurationHandler
        {
          (textField)  in
          textField.placeholder  =  "Server";
          textField.text  =  “https://sandbox.fullarmorhapi.com”;
      }
      …



   @implementation  HAPIClientBase

   - (BOOL) Login:(NSString  *)serverUrl userName:(NSString  *)userName password:(NSString  *)password
```

```
{
    self.BaseURL  = [[NSURL  alloc]  initWithString:serverUrl];

    NSString  * serverURL = [NSString  stringWithFormat:@"%@/%s",  self.BaseURL,  "route/hapi/login"];

    NSMutableDictionary  *postDict = [NSMutableDictionary  dictionary];
   [postDict  setValue:userName  forKey:@"UserName"];
   [postDict  setValue:password  forKey:@"Password"];

    NSError  *error =  nil;
    NSData* jsonData =
[NSJSONSerialization  dataWithJSONObject:postDict  options:kNilOptions  error:&error];
    NSURLResponse  *response;
    NSData  *localData =  nil;

    NSMutableURLRequest  *request =
[NSMutableURLRequest  requestWithURL:[NSURL  URLWithString:serverURL]];
   [request  setHTTPMethod:@"POST"];
[request  setHTTPMethod:@"POST"];

   [request  setValue:@"application/json"  forHTTPHeaderField:@"Content-Type"];
   [request  setValue:@"application/json"  forHTTPHeaderField:@"Accept"];
   [request  setHTTPBody:jsonData];

    // Send the request and get the response
   localData =
[NSURLConnection  sendSynchronousRequest:request  returningResponse:&response  error:&error];

    // Cookies
    self.cookies  = [[  NSHTTPCookieStorage  sharedHTTPCookieStorage  ]
              cookiesForURL:  self.BaseURL  ];

    NSString  *result = [[NSString  alloc]  initWithData:localData  encoding:NSASCIIStringEncoding];
    NSLog(@"Post result : %@", result);

    NSData  *data = [result  dataUsingEncoding:NSUTF8StringEncoding];

    NSDictionary* resultDict = [NSJSONSerialization  JSONObjectWithData:data
                                                  options:kNilOptions
                                                    error:&error];

    NSNumber  * isSuccessNumber = (NSNumber  *)[resultDict  objectForKey:  @"Success"];
    if([isSuccessNumber  boolValue] ==  YES)
   {
       self.Token  = [resultDict  objectForKey:@"Token"];
       self.UserSID  = [resultDict  objectForKey:@"UserSID"];
       self.EmailAddress  = [resultDict  objectForKey:@"EmailAddress"];
       return  true;
   }
   else
      return  false;
}
```

View Files and Folders

HAPI Calls: Shares (/route/hapi/Shares/GetFilesAndFolders) Post and Share (/route/hapi/Share/GetFilesAndFolders) Post

HAPI includes many file providers (Box, Dropbox, Office365, SharePoint, Active Directory, My Computer, etc.). The GetFilesAndFolders method for all of them are the same. The call allows for filtering, flat multi-level enumeration, and other functionality. For this sample app, we just call to get the current folder's files and folders.

We start with calling the Shares provider to get the root shares published by Active Directory. Then as the user selects to enter a share, we call the Share provider to get the file and folders for that specific folder.

Here is code from the sample app performing file and folder enumeration.

```
// Helper Method for calling HAPI REST service
func CallRestService(urlSuffix:String, fileIdentifier:String, filters:String,
        maxLevels:String) -> AnyObject
    {
        var postData: NSMutableDictionary = ["FileIdentifier":fileIdentifier,
                 "Filters":filters, "MaxLevels":maxLevels];

        var resultData = HAPIClient.sharedInstance.HttpPost(urlSuffix,
                 postDict: postData);

        var parseError: NSError?
        let parsedObject: AnyObject =
                 NSJSONSerialization.JSONObjectWithData(resultData,
                     options: NSJSONReadingOptions.AllowFragments,
                              error:&parseError)!

        return parsedObject["Items"] as AnyObject!;
    }

    // HAPI Methods
    func GetShares() -> [AnyObject]
    {
        return CallRestService("route/hapi/Shares/GetFilesAndFolders",
         fileIdentifier: "", filters: "[]", maxLevels: "0") as! [AnyObject];
    }

    func GetShare(fileIdentifier : String) -> [AnyObject]
    {
        return CallRestService("route/hapi/Share/GetFilesAndFolders",
         fileIdentifier: fileIdentifier, filters: "[]", maxLevels: "0")
         as! [AnyObject];
    }

    - (NSData*) HttpPost:(NSString *)urlSuffix postDict:(NSMutableDictionary*)
postDict
    {
        NSString * serverURL = [NSString stringWithFormat:@"%@/%@", self.BaseURL, 
urlSuffix];
        NSLog(@"%@", serverURL);

        NSString * gwToken = [self.cookies containsObject:@"GW_HAPITokenid"] ? 
[self.cookies valueForKey:@"GW_HAPIToken"] :@"";
        NSString * gwSPToken = [self.cookies containsObject:@"GW_DefaultSPSite"] ? 
[self.cookies valueForKey:@"GW_DefaultSPSite"] :@"";


        [postDict setValue:self.Token forKey:@"HAPIToken"];
        [postDict setValue:gwToken forKey:@"GW_HAPIToken"];
        [postDict setValue:gwSPToken forKey:@"GW_DefaultSPSite"];

        NSError *error = nil;
        NSData* jsonData = [NSJSONSerialization dataWithJSONObject:postDict 
options:kNilOptions error:&error];
        NSURLResponse *response;
        NSData *localData = nil;

        NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[NSURL 
URLWithString:serverURL]];
        [request setHTTPMethod:@"POST"];

        [request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
        [request setValue:@"application/json" forHTTPHeaderField:@"Accept"];
        [request setHTTPBody:jsonData];

        // Send the request and get the response
        localData = [NSURLConnection sendSynchronousRequest:request 
returningResponse:&response error:&error];

        // Cookies
        self.cookies = [[ NSHTTPCookieStorage sharedHTTPCookieStorage ]
                        cookiesForURL: self.BaseURL ];

        NSString *result = [[NSString alloc] initWithData:localData 
encoding:NSASCIIStringEncoding];

        NSData *data = [result dataUsingEncoding:NSUTF8StringEncoding];

        return data;
    }
```

Download a File

HAPI Calls: Share (/route/hapi/Share/Download) Get

The HAPI providers all have a Download method. It takes a string containing the file identifier and returns a stream filled with the file content. The stream has a ContentDispositionHeader of type attachment with a value that is the filename. Using NSURLSession can achieve the download.

Here is code from the sample app performing file download.

```
let baseUrl = serverUrl + "/route/hapi/Share/Download?fileIdentifier="
let fileIdentity = self.items[indexPath.row]["FileIdentifier"] as! String!
 fileUrl = baseUrl + fileIdentity.stringByAddingPercentEscapesUsingEncoding(NSUTF8StringEncoding)!
downloadFileWithProgress(fileUrl);
```

```
func downloadFileWithProgress(url: NSURL) {
  let session = NSURLSession(configuration:
        NSURLSessionConfiguration.defaultSessionConfiguration(),
        delegate: self, delegateQueue: nil)

  let downloadTask = session.downloadTaskWithURL(url)

        ShowProgress()
        downloadTask.resume()
 }
```

Upload a File

HAPI Calls: Share (/route/hapi/Share/UploadFile) Post

The HAPI providers all have an UploadFile method. It takes a Mime multipart request with the file identifier and chunking parameters. The HAPI Upload method can upload the file all in one call or in chunks. The sample code uploads the file in chunks.

Here is code from the sample app performing file upload.

```
    func UploadFile(serverUrl: NSURL, filePath: NSURL, uploadFolder: String!)
    {
        let nsdata = NSData(contentsOfFile: filePath.path!)!

        UploadNSData(serverUrl, nsdata: nsdata, uploadFolder: uploadFolder,
                  fileName: filePath.lastPathComponent!)
    }

    func UploadNSData(serverUrl: NSURL, nsdata: NSData!, uploadFolder: String!,
                          fileName: String!)
    {
        let bytesPerChunk = 1024*1024
        let numberOfChunk = Int(ceil(Double(nsdata.length) / Double(bytesPerChunk)))

        var i : Int
        var length : Int
        var totalLength = nsdata.length
        let mimeType = mimeTypeForPath(fileName)
        for i = 0; i < numberOfChunk; ++i {
            var lastChunk: Bool = false
            if i == (numberOfChunk - 1) {
                lastChunk = true
                length = totalLength
            }
            else
            {
                length = bytesPerChunk
                totalLength -= length
            }

            var offset : Int = i * bytesPerChunk
            var dataRange = NSRange(location: offset, length: length)

            var chunk: Int
            if numberOfChunk == 1 {
                chunk = 0
            } else {
                chunk = i + 1
            }

            var buffer : NSData = nsdata.subdataWithRange(dataRange)
            UploadFileInternal(serverUrl, fileName: fileName,
                 uploadFolder: uploadFolder, chunk: chunk, offset: offset,
                 length: length, lastChunk: lastChunk, mimeType: mimeType,
                 data: buffer);
        }
    }

    func UploadFileInternal(serverurl: NSURL, fileName: String, uploadFolder: String!,
          chunk: Int, offset: Int, length: Int, lastChunk: Bool,
         mimeType: String!, data: NSData)
    {
        let cachePolicy = NSURLRequestCachePolicy.ReloadIgnoringLocalCacheData
        var request = NSMutableURLRequest(URL: serverurl, cachePolicy: cachePolicy,
                           timeoutInterval: 2.0)
        request.HTTPMethod = "POST"

        // Set Content-Type in HTTP header.
        let boundaryConstant = generateBoundaryString()
        let contentType = "multipart/form-data; boundary=" + boundaryConstant
        //let mimeType = mimeTypeForPath(filePath.path!)

        request.setValue(contentType, forHTTPHeaderField: "Content-Type")
        request.setValue(self.Token, forHTTPHeaderField: "HAPIToken")

        // Set data
        let body = NSMutableData()
        body.appendString("--\(boundaryConstant)\r\n")
        body.appendString("Content-Disposition: form-data;
name=\"fileidentifier\"\r\n\r\n\(uploadFolder)")
        body.appendString("\r\n--\(boundaryConstant)\r\n")
        body.appendString("Content-Disposition: form-data;
name=\"chunk\"\r\n\r\n\(chunk)")
        body.appendString("\r\n--\(boundaryConstant)\r\n")
        body.appendString("Content-Disposition: form-data;
name=\"chunkStart\"\r\n\r\n\(offset)")
        body.appendString("\r\n--\(boundaryConstant)\r\n")
        body.appendString("Content-Disposition: form-data;
name=\"collisionBehavior\"\r\n\r\n1")
        body.appendString("\r\n--\(boundaryConstant)\r\n")
        body.appendString("Content-Disposition: form-data;
name=\"totalLength\"\r\n\r\n\(length)")
        body.appendString("\r\n--\(boundaryConstant)\r\n")
        body.appendString("Content-Disposition: form-data;
name=\"lastChunk\"\r\n\r\n\(lastChunk)")
        body.appendString("\r\n--\(boundaryConstant)\r\n")
        body.appendString("Content-Disposition: form-data;
name=\"file\";filename=\"\(fileName); \"\r\n")
        body.appendString("Content-Type: \(mimeType)\r\n\r\n")
        body.appendData(data)

        body.appendString("\r\n")
        body.appendString("--\(boundaryConstant)--\r\n")

        request.HTTPBody = body

        // Make an asynchronous call so as not to hold up other processes.
        NSURLConnection.sendAsynchronousRequest(request, queue:
NSOperationQueue.mainQueue(), completionHandler: {(response, dataObject, error) in
            if let apiError = error {
                self.UploadResult(error, success: false)
            } else {
                self.UploadResult(dataObject, success: true)
            }
        })
```