HAPI Android SDK Tutorial

Description

The Android SDK provides a sample java Android app using HAPI to authenticate and access on-premise Active Directory files and folders from an Android 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 Android app is written in java 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. This app demonstrates a small subset of HAPI functionality. The app was written in java using the latest Android SDK and Android Studio 1.2.2. 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.

```
public DomainLoginResponse Login(String username, String password)throws IOException,
InvalidParameterException {
    InputStream is = null;
    HttpURLConnection conn = null;
    Log.v(HAPIHELPER_TAG,"Starting Login - username=" + username);
    // check parameters
    if ( username == null || username.isEmpty() )
        throw new
InvalidParameterException(getString(R.string.error_username_required));
    if ( password == null || password.isEmpty() )
        throw new
InvalidParameterException(getString(R.string.error_password_required));
    DomainLoginResponse result = new DomainLoginResponse();
    int response = 0;
    result.Success= false;
    try {
        String myurl = gatekeeperBaseUrl + "/route/hapi/login";
        URL url = new URL(myurl);
        conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(60000 /* milliseconds */);
        conn.setConnectTimeout(45000 /* milliseconds */);
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type","application/json");
        conn.setDoInput(true);
        conn.setDoOutput(true);

        OutputStream body = conn.getOutputStream();
        JsonWriter w = new JsonWriter(new OutputStreamWriter(body,"UTF-8"));
        w.beginObject();
        w.name("UserName");
        w.value(username);
        w.name("Password");
        w.value(password);
        w.endObject();
        w.close();
        // Starts the query
        conn.connect();
        response = conn.getResponseCode();
        is = conn.getInputStream();

        // parse the response from the server
        if ( response == 200 ) {

            Reader reader = new InputStreamReader(is, "UTF-8");

            GsonBuilder builder = new GsonBuilder();
            Gson json = builder.create();
            result = json.fromJson(reader, DomainLoginResponse.class);
            Log.i(HAPIHELPER_TAG,"Login response received from server.");
            return result;
        }…}}
```

Note: For all code snippets, the gatekeeperBaseUrl variable is set to the default for the sandbox ("https://sandbox.fullarmorhapi.com").

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.

```
public FolderEnumResponse GetFilesAndFolders(FolderEnumRequest request)throws
IOException, InvalidParameterException {
    InputStream is = null;
    HttpURLConnection conn = null;
    Log.v("HAPIHelper","GetFilesAndFolders Starting");
    if ( request == null ) {
        throw new InvalidParameterException(getString(R.string.error_null_request));
    }
    try {
        String providerName = "Shares";
        if (request.FileIdentifier != null && !request.FileIdentifier.isEmpty())
            providerName = "Share";
        String myurl = gatekeeperBaseUrl + "/route/hapi/" + providerName +
"/GetFilesAndFolders";
        URL url = new URL(myurl);
        conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(60000 /* milliseconds */);
        conn.setConnectTimeout(45000 /* milliseconds */);
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setRequestProperty("HAPIToken", this.mToken);
        conn.setDoInput(true);
        conn.setDoOutput(true);

        OutputStream body = conn.getOutputStream();
        Gson gBody = new Gson();
        String jsonString = gBody.toJson(request);
        OutputStreamWriter bodyWriter = new OutputStreamWriter(body);
        bodyWriter.write(jsonString);
        bodyWriter.close();
        conn.connect();
        int response = conn.getResponseCode();


        if (response == 401) {
            Log.e(HAPIHELPER_TAG,"GetFilesAndFolders::401 error - redirecting to
login");
            DoLogon();
            return null;
        }

        // parse the response from the server
        if (response == 200) {

            is = conn.getInputStream();
            Reader reader = new InputStreamReader(is, "UTF-8");

            GsonBuilder builder = new GsonBuilder();
            Gson json = builder.create();
            FolderEnumResponse result = json.fromJson(reader,
FolderEnumResponse.class);
            if ( result != null && result.Items != null )
                Log.d(HAPIHELPER_TAG,"GetFilesAndFolders returned " +
String.valueOf(result.Items.length));
            return result;
        }…}}
```

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.

Here is code from the sample app performing file download.

```
public Uri DownloadFileString fileIdentifier,Context context)throws IOException,
InvalidParameterException {
    InputStream is = null;
    HttpURLConnection conn = null;
    Log.v("HAPIHelper","Downloading file " + fileIdentifier);
    if ( fileIdentifier==null || fileIdentifier.isEmpty() )
        throw new
InvalidParameterException(getString(R.string.error_fileidentifier_required));
    try {
        String providerName = "Share";
        String myurl = gatekeeperBaseUrl + "/route/hapi/" + providerName +
"/Download";
        myurl += "?fileIdentifier=" + URLEncoder.encode(fileIdentifier);
        URL url = new URL(myurl);
        conn = (HttpURLConnection) url.openConnection();
        conn.setConnectTimeout(45000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setRequestProperty("HAPIToken", this.mToken);
        conn.setDoInput(true);

        // get the file name from the fileIdentifier
        String fileName = fileIdentifier;
        int ndx = fileIdentifier.lastIndexOf('\\');
        if ( ndx > 0 ) {
            fileName = fileIdentifier.substring(ndx+1);
        }
        int response = conn.getResponseCode();


        if (response == 401) {
            Log.e(HAPIHELPER_TAG,"Download::401 error - redirecting to login");
            // TODO: set context so we can retry the operation after logging in again?
            DoLogon();
            return null;
        }

        // parse the response from the server
        if ( response == 200 ) {
            is = new BufferedInputStream(conn.getInputStream());

            // this will be useful to display download percentage
            // might be -1: server did not report the length
            int fileLength = conn.getContentLength();
            // download the file
            boolean bOK = isExternalStorageWritable();
            File targetFolder =
context.getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS);
            File targetFile = new File(targetFolder,fileName);


            FileOutputStream os = new FileOutputStream(targetFile);

            byte data[] = new byte[4096];
            long total = 0;
            int count;
            while ((count = is.read(data)) != -1) {
                // allow canceling with back button
                if (isCancelled()) {
                    is.close();
                    return null;
                }
                total += count;
                // publishing the progress....
                if (fileLength > 0) { // only if total length is known
                    // publishProgress((int) (total * 100 / fileLength));
                    os.write(data, 0, count);
                }
            }
            os.flush();
            os.close();
            os = null;
            return Uri.fromFile(targetFile);
        }…}}
```

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.

```
public Boolean UploadFile(String hapiToken, Uri sourceFile, String targetFolder,
String targetFileName) throws IOException, ExecutionException {

    InputStream is = null;
    HttpURLConnection conn = null;
    try {
        Log.d(HAPIHELPER_TAG,"UploadFile starting.");
        // check parameters
        if ( sourceFile == null )
            throw new
InvalidParameterException(getString(R.string.error_field_required));
        String myurl = gatekeeperBaseUrl + "/route/hapi/Share/UploadFile";
        URL url = new URL(myurl);
        conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(60000 /* milliseconds */);
        conn.setConnectTimeout(45000 /* milliseconds */);
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Accept", "application/json");

        conn.setRequestProperty("HAPIToken", hapiToken);
        conn.setDoInput(true);

        File file = new File(sourceFile.getPath());
        int offset = 0;
        int chunkNum = 0;
        int chunkSize = 1*1024*1024;  // 1MB
        int currentChunkBytes = 0;
        long numBytes = file.length();
        double totalChunks = Math.ceil(numBytes/chunkSize);
        while ( offset < numBytes ) {
            MultipartRequest request = new MultipartRequest(conn);
            request.addFormField("fileidentifier", targetFolder);
            request.addFormField("chunk", String.valueOf(chunkNum));
            request.addFormField("chunkStart", String.valueOf(offset));
            request.addFormField("lastChunk", String.valueOf((chunkNum >= totalChunks)
? true : false));
            currentChunkBytes = request.addFilePartChunk("file",
file,offset,chunkSize);
            request.finish();
            conn.connect();
            int response = conn.getResponseCode();
            if (response == 200) {
                // the upload succeeded
                Log.d(HAPIHELPER_TAG,"Uploaded " + String.valueOf(currentChunkBytes) +
"bytes in chunk " + String.valueOf(chunkNum));
                chunkNum++;
                offset += currentChunkBytes;
                Log.d(HAPIHELPER_TAG, String.valueOf(offset) + "bytes of " +
String.valueOf(numBytes) + " total have been uploaded." );

            }
            else if ( response == 401 ) {
                // redirect to logon
                Log.e(HAPIHELPER_TAG, "UploadFile::401 error - redirecting to login");
                DoLogon();
                return false;
            }
            else {
                Log.e(HAPIHELPER_TAG,"UploadFile call failed. Response = " +
String.valueOf(response));
                break;
            }
        }…}}
```