HAPI Web UI Tutorial

Description

The Web UI provides a sample integration into any browser based application using HAPI to authenticate and interact with files from that application. Unlike the other sample apps, the Web UI demonstrates more functionality. It deals with shared files, goes beyond just Active Directory (AD) files to Box and Dropbox files as well, and supports file and folder copy and sync jobs.

Tutorial

Introduction

This sample Web UI code uses HTML5, JavaScript, Angular JS, and JQuery. It demonstrates how to use HAPI to login, navigate files and folder published in Active Directory, download a file, upload a file into Active Directory, view shared files, and perform two factor authorization with a SAAS-based file storage product to access its files. 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 simplified JavsScript code (based on the sample app) performing a login.

```
function TryLogin() {
     HAPILogin(_
          hapiServer, // (https://sandbox.fullarmorhapi.com)
          $("#loginUsername").val(), // get user name from UI
          $("#loginPassword").val(), // get pw from UI
          function () {              // callback.
               CloseModalBox();     // close the login prompt
               SetUserPic();        // show user pic from AD

               var url = "ajax/hapiADShares.html"; // AD Shares View
               window.location.hash = url;
               LoadAjaxContent(url); // go to default view
               });
}
```

And under the hood, token tracking, error handling…

```
function HAPILogin(server, username, password, callback) {
    var urlSuffix = "/route/hapi/login";
    var loginUrl = server + urlSuffix;
    console.log(loginUrl);

    $.ajax({
        type: 'POST',
        dataType: 'json',
        contentType: "application/json; charset=utf-8",
        url: loginUrl,
        data: JSON.stringify({ UserName: username, Password: password }),
        success: function (data) {

            $.cookie("hapiToken", data.Token, {expires: 7, path: "/"});
            $.cookie("HAPI.UserSID", data.UserSID, {expires: 7, path: "/"});
            $.cookie("HAPI.UserRole", data.Role, {expires: 7, path: "/"});

            if (callback != null)
                callback();
        },
        error: function (xhr, ajaxOptions, thrownError) {
            alert("Error: " + xhr.responseText + "\n" + thrownError);
        }
    });
}
```

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.

HTML Table using AngularJS syntax to bind to a list of AD Shares:

```
<table id="naviDataTable" class="table table-bordered table-striped table-hover table-heading
table-datatable" data-toggle="context" data-target="#context-menu">

     <thead>
          <tr>
              <th>Name</th><!–column header for Name of the share -->
              <th>Path</th><!–column header for Path of the share-->
          </tr>
     </thead>
     <tbody>
```

```
            <tr ng-repeat="item in Shares | filter: searchKeyword" ng-click="setSelected(item)"
                ng-class="{selected: item.FileIdentifier === selectedItem.FileIdentifier}">
                <td><!--cell shows Name of the share-->
                       <a class="ajax-link" href="ajax/hapiADFiles.html"
                          data-context="{{item}}" target="_blank">
                           <i class="fa fa-folder-o hapi-row-icon"></i>{{ item.Name }}
                       </a>
                </td>
                <td><!--cell shows Path of the share-->
                       {{ item.FileIdentifier }}
                </td>
            </tr>
      <tfoot />
</table>
```

Getting a list of shares using AngularJS. HTML table above binds to these results:

```
$scope.GetShares = function () {

      var theUrl = _hapiServer + "/route/hapi/Shares/GetFilesAndFolders";

      var hapiToken = $.cookie("hapiToken");
      var postData = JSON.stringify({ fileIdentifier: "", filters: "[]" });

      // Simple POST in AngularJS (passing data) :
      $http({
            url: theUrl,
            method: "POST",
            data: postData,
            headers: { 'Content-Type': 'application/json', 'hapiToken': hapiToken }
      }).
      success(function (data, status, headers, config) {
            // HTML Table binds to these results
            $scope.Shares = data.Items;
      }).
      error(function (data, status, headers, config) {
            // called asynchronously if an error occurs
            // or server returns response with an error status.
            HandleStatus(status, 'ADSharesController');
      });
};
```

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.

```
$scope.downloadItem = function(){

      var theUrl = _hapiServer + "/route/Share/Download?fileIdentifier=" +
      encodeURIComponent($scope.selectedItem.FileIdentifier);

      window.location.assign(dlurl);
}
```

Here is a sample URL for downloading a file from a Hapi server:
https://qa4.hapidevelopment.com/route/hapi/Share/Download?fileIdentifier=\\WIN-UEE5OSKNIL4\Users\tesla-model-s.pdf&HAPIToken=d24a91c7-4a27-49a1-98cd-ff24019f5852

List Shared Files

HAPI Calls: SharedFile (/api/SharedFile/GetSharedFiles) Post

The HAPI Gatekeeper allows you to create, update, list, and remove files from all providers. Creating a shared file link allows any user a link to download the file. In the Web UI sample, we illustrate all these operations.

Here is code from the sample code performing the listing of shared files.

```
$scope.GetLinks = function () {

   var urlSuffix = "/api/SharedFile/GetSharedFiles";
   _Url = _hapiServer + urlSuffix;

   var hapiToken = $.cookie("hapiToken");

   var postData = JSON.stringify({ Provider: "None", IncludeExpired: true,
   FileIdentifier_Checksum: 0, UserSID: "" });

    $('#main').css( 'cursor', 'wait' );
    // Simple POST request (passing data) :
    $http({
        url: _Url,
        method: "POST",
        data: postData,
        headers: { 'Content-Type': 'application/json', 'hapiToken': hapiToken }
    }).
      success(function (data, status, headers, config) {
          // this callback will be called asynchronously
          // when the response is available

          ToUILinksModel(data.Items);
          $scope.Links = data.Items;
          $('#main').css( 'cursor', 'default' );
      }).
      error(function (data, status, headers, config) {
          // called asynchronously if an error occurs
          // or server returns response with an error status.
          HandleStatus(status);
          $('#main').css( 'cursor', 'default' );
      });
};
```

SaaS-Based File Storage Integration

HAPI Calls: Dropbox (/route/hapi/Dropbox/GetFilesAndFolders) Post

Some of the HAPI providers allow access to SaaS-based file storage products (Dropbox, Box, Office365, etc.). HAPI uses two factor authorization to access those products and perform operations.

The following section describes how to make requests to Dropbox and how to enable your user to provide authentication when needed.

Login to Dropbox

When accessing Dropbox files through HAPI, the user may need to authenticate with Dropbox. Your code should make normal requests to HAPI for files from Dropbox. HAPI will respond to your request that external authentication is needed by returning a code 500 and will include a redirect URL to the Dropbox authentication page. Your code can read the redirect URL (from the header `HAPI-External-Auth-Required` for the code 500) as shown in the 500 handler below, and display the Dropbox login page to the user. The user will see Dropbox's own authorization page asking for username and password with an option to enable the user to access their Dropbox files through HAPI. Once the user has provided their credentials to Dropbox and granted permission for HAPI, your normal file operations for Dropbox will be authenticated and will succeed.

```
function HandleError(xhr, status) {
    switch (xhr.status) {
        case 401:
            // Need to Login to Hapi
            ShowLogin();
            return true;

        case 403:
            console.log("Forbidden: " + xhr.responseText);
            return true;

        case 404:
            alert(xhr.responseText) // Show the error to the user
            return true;

        case 500:
            var header = xhr.getResponseHeader("HAPI-Reset-Provider");
            if (header) {
                window.location.href = 'ajax/hapiADShares.html';
                return true;
            }

            // External Authentication Required
            header = xhr.getResponseHeader("HAPI-External-Auth-Required");
            if (header) {
                // Show the external authentication page to the user
                window.location.href = CreateNewRedirectDropboxUrl(header);
                return true;
            }

            // 500 not handled, show error to the user
            alert("Error:\r\n" + status + "\r\n" + JSON.stringify(xhr));
            return false;

        default:
            alert("Error:\r\n" + status + "\r\n" + xhr);
            return false;
    }

    return false;
}
```

```
function CreateNewRedirectDropboxUrl(oldUrl)
{
    pos = oldUrl.lastIndexOf('&')
    firstPart = oldUrl.substring(0, pos)
    lastPart = oldUrl.substring(pos)
    fullurl = firstPart + '|' + location.protocol + '//' + location.host + lastPart
    return fullurl
}
```

Get Files and Folders from Dropbox

```
$scope.GetFilesAndFolders = function () {
        var theUrl = _hapiServer + "/route/hapi/Dropbox/GetFilesAndFolders"; // Note: Dropbox
provider

        var hapiToken = $.cookie("hapiToken");
        var postData = JSON.stringify({ fileIdentifier: contextItem.FileIdentifier, filters:
"[]" });

        $('#main').css( 'cursor', 'wait' );
        // Simple POST request (passing data) :
        $http({
            url: theUrl,
            method: "POST",
            data: postData,
            headers: { 'Content-Type': 'application/json', 'hapiToken': hapiToken }
        }).
          success(function (data, status, headers, config) {
       // this callback will be called asynchronously
       // when the response is available
       ToUIModel(data.Items, 'Dropbox');
       $scope.FilesAndFolders = data.Items;

             $('#main').css( 'cursor', 'default' );
          }).
          error(function (data, status, headers, config) {
       // called asynchronously if an error occurs
       // or server returns response with an error status.
       HandleStatus(status, "DropboxController", headers);
       $('#main').css( 'cursor', 'default' );
          });
    };
```