

Using REST hook to authorize user by domain

Overview

Let's suppose that there is task to limit user access to server content on domain basis, e.g. allow users to watch a video from specified domain only. This task can be resolved with REST hooks.

In REST query of type 1 `connect` to the backend server, WCS sends the `origin` field that contains WCS server domain name by which user accessed to it, for example

```
{
  "nodeId" : "5tWOFn5ZoMQs22KrEls2Ulhee57hQ09D",
  "appKey" : "defaultApp",
  "sessionId" : "/192.168.1.45:53438/abcdef0123456789",
  "useWsTunnel" : false,
  "useWsTunnelPacketization2" : false,
  "useBase64BinaryEncoding" : false,
  "mediaProviders" : [ "WebRTC", "MSE", "WSPlayer" ],
  "clientVersion" : "0.5.28",
  "clientOSVersion" : "5.0 (Windows)",
  "clientBrowserVersion" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101 Firefox/60.0",
  "keepAlive" : false,
  "origin" : "https://test2.flashphoner.com:8888"
}
```

So, REST hook `/connect` should be implemented to authorize user by domain.

The REST hook of type 1 implementation example to authorize by domain

Implementation requirements

1. REST hook should be available on your web host as `http://yourhost/rest-hooks/connect`
2. REST hook should allow POST `application/json` HTTP requests
3. REST hook should return exactly the same body as received (mirroring), except `restClientConfig` configuration
4. WCS server should be set up to work with REST hook as:

```
ssh -p 2001 admin@localhost
>update app -l http://yourhost/rest-hooks defaultApp
```

REST hook using can be traced in [WCS server logs](#)

```
tail -f /usr/local/FlashphonerWebCallServer/logs/server_logs/flashphoner.log
```

The code of the example and its analysis

The first lines of the script define the method and decode the request body. Here the domain for authorization is set:

```
<?php
$api_method = array_pop(explode("/", $_SERVER['REQUEST_URI']));
$incoming_data = json_decode(file_get_contents('php://input'), true);
$domain = "yourdomain.com";
```

`connect` method handling begins. Here `origin` field value is defined and `restClientConfig` field for answer is filled:

```
switch($api_method) {
    case"connect":

        $origin = $incoming_data['origin'];

        //logs
        error_log("sessionId: " . $incoming_data['sessionId']);
        error_log("origin: " . $origin);

        $rest_client_config =
        json_decode(file_get_contents('rest_client_config.json'), true);
        $incoming_data['restClientConfig'] = $rest_client_config;
        ...
}
```

Domain checking. If domain is not found, the `ubnormalResponse()` function is called to form negative response 403:

```
switch($api_method) {
    case"connect":
        ...
        $found = strpos($origin, $domain);

        if ($found !== false){
            error_log("User authorized by domain " . $domain);
        } else {
            error_log("User not authorized by domain: " . $domain . "
Connection failed with 403 status.");
            ubnormalResponse(403);
        }
}
```

```
    }  
    break;  
    ...  
}
```

Successful response is returned

```
header('Content-Type: application/json');  
echo json_encode($incoming_data);
```

`ubnormalResponse()` function completes the script:

```
function ubnormalResponse($code) {  
    if ($code == 403) {  
        header('HTTP/1.1 403 Forbidden', true, $code);  
    } else {  
        header(':', true, $code);  
    }  
    die();  
}  
?>
```

 [Full script example](#) >

 `restClientConfig` settings from file `rest_client_config.json` for response to `/connect` query >