# HTTP Requests

|  |  |
| --- | --- |
| content-type | application/json; charset=UTF=8 |
| method | POST |
| url | {**url**}/layouts/TaugorGED17/Services/MainController.aspx/ExecuteAction |
| body/data/payload | {  "controller":"Authentication",  "action":"{**action** : string}",  "parameters”: {**parameters** : object},  "username":null,  "requestToken":null  } |

WHERE

**url** = Taugor GED’s URL

**action** = action to execute, according to the steps below

**parameters** = parameters for the given action, according to the steps below

# Step 1 – Handshake

## Returns

question: string

Arguments:

**action** = "Handshake"

**parameters** = {

"credentials":{

"UserName": {**username**: string},

"Password": {**password**: string}

},

"subject": {**subject**: string}

}

WHERE

**username** = user’s account

**password** = user account’s password

**subject** = eg.: "SSO"

# Step 2 – Authenticate

## Returns

{

"User": {

LoginName: string,

[…]

}

"RequestToken": string

}

Arguments:

**action** = "Authenticate"

**parameters** = {

"credentials":{

"UserName": {username: string},

"Password": {password: string}

},

"subject": {string}

"fingerprint": {fingerprint: string}

}

WHERE

**username** = user’s account (same as in Step 1)

**password** = user account’s password (same as in Step 1)

**subject** = as given by Taugor (same as in Step 1)

**fingerprint** = answer to the handshake’s question as instructed by Taugor

# Códido de Exemplo de utilização

//Here's a simple JavaScript implementation for abstracting the http request using axios

/\*

function http(url, controller, action, parameters){

return axios({

method: 'POST',

url: url + '/\_layouts/TaugorGED17/Services/MainController.aspx/ExecuteAction',

headers: {

'Content-Type': 'application/json; charset=utf-8',

},

data: JSON.stringify({

controller: controller,

action: action,

parameters: parameters || null,

username: null,

requestToken: null,

}),

});

}

\*/

//Dummy validation for the presentation

const SuperHeroLookalikes = {

redtornado: 'vision'

}

//url for your Taugor GED environment

//(this is a development environment without DNS)

const gedUrl = 'http://186.223.228.136:5003'

//These credentials should be for the user which you want to authenticate

const credentials = { userName: 'Administrator', password: 'Absirwt600' }

//The following method used, API.actionCamelCase is an abstraction for the Http Request like the one above,

//with an additional step for getting only the data and transforming the fields to camel case hence the server returns in pascal case

//Start with a handshake and get the question

API.actionCamelCase('Authentication', 'Handshake', {

credentials,

subject: 'SmartECM'

}).then(question => {

//As you receive the question, answer it to complete your authentication

API.actionCamelCase('Authentication', 'Authenticate', {

credentials,

subject: 'SmartECM',

fingerprint: SuperHeroLookalikes[question]

}).then(r => {

//Finally, send the use to the following URL

const url = `${gedUrl}/\_layouts/TaugorGED17/App/Authenticate.aspx?username=${r.user.loginName}&token=${r.requestToken}`

window.location.href = url

})

})

# UTILIAÇÃO PARA CADASTRO DE USUÁRIO

# HTTP Requests

|  |  |
| --- | --- |
| content-type | application/json; charset=UTF=8 |
| method | POST |
| url | {**url**}/layouts/TaugorGED17/Services/MainController.aspx/ExecuteAction |
| body/data/payload | {  "controller":"**{ controler**: string} ",  "action":"{**action** : string}",  "parameters”: {**parameters** : object},  "username":null,  "requestToken":null  } |

# Step 2 – CreateUser

Method to create the user in the GED

## Returns

question: string

Arguments:

**action** = "CreateUser"

**parameters** = {

"user":{

"Name": {**name**: string},

"LastName": {**LastName**: string}

"LoginName": {**LoginName**: string}

"Password ": {**Password**: string}

"Type": { **Type**: string } // Type : “External”, “Basic”, “Full”

"Enabled": { **Enabled**: bool}

"Email": { **Email**: string}

"NotifyOnFirstLogin": { **NotifyOnFirstLogin**: bool}

"SendWelcomeEmail": { **SendWelcomeEmail**: bool}

"EmailBody": { **EmailBody**: string}

},

}

WHERE

**name** = user’s account

**LastName** = user account’s name

**LoginName** = user account’s name login

**Password** = user account’s passord

**Type** = Type : “External”, “Basic”, “Full”

**Enabled** = user account’s enabled

**Email** = user account’s email

**NotifyOnFirstLogin**= user account’s notify where on first login

**SendWelcomeEmail** = user account’s send welcome email?

**EmailBody** = user account’s – email body

**Example code:**

One must first get the Workspace, then Organizational Unit, then Enterprise, and finally the Enterprise group that he has permission in the GED.

user = Principals.CreateUser(NewUserVM user) : User

workspaces = Query.Get(query) ou Workspaces.GetAll(query)

ws = [...]

Workspaces.AddMembers(ws.id, user.id)

organizationalUnits = Query.Get(query) ou Workspaces.GetAll(query)

ou = [...]

OrganizationalUnit.AddMembers(ou.id, user.id)

ou.Companies

comp = [...]

groups = Query.Get(query) //CompanyGroups

Company.AddMember(CompanyGroupCtrlVM group) //title, companyIdentifier, Members {id: X}[]

Company.RemoveMember(CompanyGroupCtrlVM group) //title, companyIdentifier, Members {id: X}[]

# Step 3 – Lock User

Principal.Lock(string loginName)

Principal.Unlock(string loginName)

**Super Hero Lookalikes**

|  |  |
| --- | --- |
| Pergunta (Handshake) | Resposta (Authenticate) |
| deathstroke | deadpool |
| bigbarda | gamora |
| redhood | wintersoldier |
| swampthing | manthing |
| greenarrow | hawkeye |
| aquaman | namor |
| elongatedman | mrfantastic |
| atom | antman |
| redtornado | vision |
| doompatrol | xmen |
| catwoman | blackcat |
| greenlantern | nova |
| batman | moonknight |
| wonderwoman | powerprincess |
| superman | hyperion |