Administration (scripting)

Custom scripts

Introduction to Custom scripts

What is a Custom script?

The Administration Addon provides the capacity to launch custom scripts to perform any functionality or process that the Soffid API has available.

Additionally, with this addon, there is available the possibility to enable a special to get the metrics of the performance of the Soffid IAM components.

Through the Custom scripts page you could perform the next operations:

- You could create a new custom script.
- You could execute the custom script **On demand** by clicking on the Execute now button.
- You could execute the custom script as a **Scheduled** task, and it will be executed as you have configured the timetable. *(*1)*
  - **Month**: number of the month (1-12) when the task will be performed.
  - **Day**: number of the day (1-31) when the task will be performed.
  - **Hour**: hour (0-23) when the task will be performed.
  - **Minute**: minute (0-59) when the task will be performed.
  - **Day of week**: number of day (0-7 where 0 means Sunday) of the week when the task will be performed.
  - **Server**: where the agent is running.
- You could define the event in which the script will be launched:
  - **On user change**: this kind of script will be launched when any user changes.
  - **On revoke permission**: this type of script will be launched when permissions are revoked.
  - **On grant permission**: this type of script will be launched when permissions are assigned.

On user change

When you define an event on user change, the user object will be available on the **user variable** to access and operate with it.

To find more details about the user object, you can visit theSoffid Objects page.

On grant permission

When you define an event on revoke or grant permission, the grant object will be available on the **grant variable** to access and operate with it.

To find more details about the grant object, you can visit the Soffid Objects page.

Documentation

Below you could find a list of helpful links related to the building of custom scripts

API for the internal classes of Soffid: https://download.soffid.com/doc/console/latest/uml/

Custom utility classes: https://bookstack.soffid.com/books/soffid-3-reference-guide/page/utility-classes

---

*(*1) For each value of mont, day, hour, minute or day of the week:*

- ** means any month, day, hour, minute, or day of week. e.g. */5 to schedule every five minutes.*
- *A single number specifies that unit value: 3*
- *Some comma separated numbers: 1,3,5,7*
- *A range of values: 1-5*

How to install Custom scripts in Soffid

Installation

Download

Please download the Soffid Administration (scripting) add-on.

You can download it at the following link http://www.soffid.com/download/enterprise/ if you have Soffid user with authorization, or in the following http://download.soffid.com/download/ by registering.

Upload

1. Once the Federation add-on is downloaded, please log in to IAM Console.

> You need to be an administrator user of the Soffid console or a user with permissions to upload addons.

> It is recommended to upload the addons to master, this is the way to maintain updated all, master and tenants if there are.

2. In the Soffid console, please go to:

```
Main Menu > Administration > Configure Soffid > Global Settings > Plugins
```

3. Then, click the add button (+) and pick the file and Soffild will upload the addon file.

> For more information visit the Addons Getting started page.

4. Finally, when the addon is installed, it will be required to restart the Soffid Console.

5. Once the Soffid console is restarted, you could check the plugin was uploaded properly on the plugins page:

```
Main Menu > Administration > Configure Soffid > Global Settings > Plugins
```

6. Now, you can use the Custom scripts.

Custom scripts samples

Custom scripts samples

Sample scripts

> Note that Soffid supports different scripting languages, you can configure it in the Smart engine settings screen.

> Additionally, in the initial configuration of the container, we can configure the SOFFID_TRUSTED_SCRIPTS environment variable to allow the use of insecure classes. You can find this information visiting the Installing IAM Console page.

Table of contents

Agent scripts

User full name

```
return firstName + lastName;
```

Create mainDomain if it doesn't exit

```
String mailDomain = null;
if (email != void && email != null && email.contains("@")) {
    String[] mailTokens = email.split("@");
    mailDomain = mailTokens[1];
}
com.soffid.iam.service.MailListsService service =
com.soffid.iam.ServiceLocator.instance().getMailListsService();
com.soffid.iam.api.MailDomain domain = service.findMailDomainByName(mailDomain);
if (domain==null) {
    domain = new com.soffid.iam.api.MailDomain();
    domain.setCode(mailDomain);
    domain.setDescription(mailDomain);
    domain.setObsolete(new Boolean(false));
    domain = service.create(domain);
```

```
}
return mailDomain;
```

Recover active agents

```
llistaAgents =  serviceLocator.getDispatcherService().findAllActiveDispatchers();
for(agent:llistaAgents) {
  out.println("Nom: " + agent.name);
  out.println("Class Name: " + agent.className + "\n");
}
```

Show by a user the agents that have associates

```
llistaUsuaris = serviceLocator.getUserService().findUserByJsonQuery("userName eq \"Ivan\" ");
for(usuari:llistaUsuaris) {
  out.println("Usuario: " + usuari.userName);

   llisstacuentas =
serviceLocator.getAccountService().findAccountByJsonQuery("users.user.userName eq
\""+usuari.userName+"\" ");

  for(cuenta:llisstacuentas){
    out.print("   Cuenta : " + cuenta.name);
    out.println("   ID: " + cuenta.id);
    llistaRole = serviceLocator.getApplicationService().findRoleAccountByAccount(cuenta.id);

    for(role:llistaRole){
      out.print("      Role: " + role.roleName + "\n");
      out.println("          Aplicacion: " + role.informationSystemName);
      out.println("             Agente: " + role.system);
    }
  }
}
```

Identity scripts

Recover a user for userName

```
u = serviceLocator.getUserService().findUserByUserName("Ivan");
out.print("Usuari: " + u.firstName);
```

Recover a users from a Jquery

```
llistaUsuari = serviceLocator.getUserService().findUserByJsonQuery("firstName sw \"A\" AND
lastName sw \"V\" ");
for (usuari:llistaUsuari){
    out.println("Usuari: " + usuari.userName);
}
```

Print some attributes

```
u = serviceLocator.getUserService().findUserByUserName("02");
out.println("UserName: " + u.userName);
out.println("Name: " + u.firstName);
out.println("LastName: " + u.lastName);
```

Print by user the email

```
u = serviceLocator.getUserService().findUserByUserName("02");
out.print("Email: " + u.shortName + "@" + u.mailDomain);
```

Print by user some additional data

```
llistaDadesUsuari = serviceLocator.getUserService().findUserDataByUserName("18008366X");
for(dadaUsuari:llistaDadesUsuari){
  out.println("Atributs " + dadaUsuari.attribute + " = " + dadaUsuari.value);
}
```

Create a new identity

```
 try {
 newUser = new com.soffid.iam.api.User();
//Instanciar un nuevo objeto de tipo usuario

 newUser.userName = "IvanVis"; //Faltan 6 parametres
 newUser.firstName = "Ivannn";
 newUser.lastName = "Visarttt";
 newUser.userType = "I";
```

```
 newUser.profileServer = "null" ;
 newUser.homeServer = "null" ;
 newUser.mailServer = "null" ;
 newUser.primaryGroup = "world";
 newUser.active = true;

 serviceLocator.getUserService().create(newUser);
}catch(Exception e){
 e.printStackTrace(out);
}
```

Update an identity

```
u = serviceLocator.getUserService().findUserByUserName("Ivan");
u.firstName = "Ivaaan1";
u = serviceLocator.getUserService().update(u);
out.print(u.firstName);
out.print(u.userName);
```

Delete an identity

```
try {
  u = serviceLocator.getUserService().findUserByUserName("02");
  serviceLocator.getUserService().delete(u);
} catch(Exception e) {
    e.printStackTrace(out);
}
```

Account scripts

Recover accounts of user

```
la = serviceLocator.getAccountService().findAccountByJsonQuery("users.user.userName eq \"02\"
");
for(a:la) {
  out.println("Cuenta: " + a.name);
  out.println("ID: " + a.id);
```

```
  out.println("System: " + a.system + "\n");
}
```

Remove attribute values of a metadata

```
public void removeUnAttributeValues(String attribute, String system) {
  la = serviceLocator.getAccountService().findAccountByJsonQuery("system eq \""+system+"\"");
  for (a : la) {
    laa = serviceLocator.getAccountService().getAccountAttributes(a);
    for (aa : laa) {
      if (aa.attribute.equals(attribute)) {
        if (aa.value!=null) {
          out.print("accountName: "+accountName+", attribute.value: "+aa.value);
          serviceLocator.getAccountService().removeAccountAttribute(aa);
          out.println(" ---> removed");
        }
      }
    }
  }
}
removeUnAttributeValues("manager","OSCM");
```

Role scripts

Recover roles of a user

```
user = serviceLocator.getUserService().findUserByUserName("Ivan");
out.println("Usuari: " + user.userName + "\n");
rolsUser = serviceLocator.getUserService().findUserRolesHierachyByUserName(user.userName);
for(listrRolsUser:rolsUser){
  out.println("Nombre: " + listrRolsUser.name);
  out.println("Descripcion: " + listrRolsUser.description);
  out.println();
}
```

Print the associated roles for each account

```
llistaUsuaris = serviceLocator.getUserService().findUserByJsonQuery("userName eq \"Ivan\" ");
for(usuari:llistaUsuaris){

   llisstacuentas =
serviceLocator.getAccountService().findAccountByJsonQuery("users.user.userName eq
\""+usuari.userName+"\" ");

  for(cuenta:llisstacuentas){
    out.print("    Cuenta : " + cuenta.name);
    llistaRole = serviceLocator.getApplicationService().findRoleAccountByAccount(cuenta.id);

    for(role:llistaRole){
      out.print("       Role:  " + role.roleName + "\n");
    }
  }
}
```

Print for an account the roles and applications for each of them

```
llistaUsuaris = serviceLocator.getUserService().findUserByJsonQuery("userName eq \"Ivan\" ");
for(usuari:llistaUsuaris){

   llisstacuentas =
serviceLocator.getAccountService().findAccountByJsonQuery("users.user.userName eq
\""+usuari.userName+"\" ");

  for(cuenta:llisstacuentas){
    out.print("    Cuenta : " + cuenta.name);
    out.println("    ID:  " + cuenta.id);
    llistaRole = serviceLocator.getApplicationService().findRoleAccountByAccount(cuenta.id);

    for(role:llistaRole){
      out.print("       Role:  " + role.roleName + "\n");
      out.println("           Aplicacion:  " + role.informationSystemName);
    }
  }
}
```

Print the roles associated with each account

```
usuCuenta = serviceLocator.getUserService().findUserByJsonQuery("");
for(listaUsuCuenta:usuCuenta) {

    out.println("Usuario: " + listaUsuCuenta.userName);
    out.println("Nombre: " + listaUsuCuenta.firstName);

    rolsUser =
serviceLocator.getUserService().findUserRolesHierachyByUserName(listaUsuCuenta.userName);

    for(listaRolsUser:rolsUser){
      out.println("Nombre del Rol: " + listaRolsUser.name);
      out.println("Descripcion: " + listaRolsUser.description);
      out.println();
    }
  }
}
```

Create a new role

```
try {
  newRol = new com.soffid.iam.api.Role();
  newRol.name = "Rol_New_Script";
  newRol.description = "Rol Script";
  newRol.informationSystemName = "SOFFID";
  newRol.system = "APLICACION01";
  serviceLocator.getApplicationService().create(newRol);

} catch(Exception e){
    e.printStackTrace(out);
}
```

Update a role

```
editRole = serviceLocator.getApplicationService().findRoleByJsonQuery("name eq \"Rol editado
por script\" and informationSystemName eq \"APLICACION01\" ");
for (role:editRole){

  out.println(role.name);
```

```
  role.name = "ROL01";

  role = serviceLocator.getApplicationService().update(role);
  out.print(role.name);
}
```

Delete a role

```
try {
  editRole = serviceLocator.getApplicationService().findRoleById(232734);
  serviceLocator.getApplicationService().delete(editRole);
} catch(Exception e){
    e.printStackTrace(out);
}
```

List the roles of an application

```
list = serviceLocator.getApplicationService().findRoleByJsonQuery("informationSystemName eq
\"SOFFID\"");
for (role : list) {
  out.println(role.name);
}
```

Event Sample scripts

On grant permission: Update a user attribute when assigning a specific permission

```
if (grant.roleName.equals("RS002")) {
  user = serviceLocator.getUserService().findUserByUserName(grant.user);
  if (user != null) {
    attributes = serviceLocator.getUserService().findUserAttributes(user.userName);
    if (attributes == null) {
        attributes = new HashMap();
    }
    attributes.put("language", "Spanish");
    serviceLocator.getUserService().updateUserAttributes(user.userName, attributes);
  }
}
```