Sample scripts

Introduction

> Note that Soffid supports different scripting languages, you can configure it in the Smart engine settings screen. **Soffid 4** configures the smart engine with **Javascript** scripting language as the default.

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.

Custom scripts page

The following **examples** of custom scripts can be run directly on the Custom script page.

These scripts can also be used in any other Soffid script component.

The scripts have been generated for the **Javascript engine**.

Identity scripts

Recover a user for userName

```
var u = serviceLocator.getUserService().findUserByUserName("admin");
out.print("User: " + u.firstName);
```

Print some attributes

```
var u = serviceLocator.getUserService().findUserByUserName("test");
out.println("UserName: " + u.userName);
```

```
out.println("Name: " + u.firstName);
out.println("LastName: " + u.lastName);
```

Print by user the email

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

Print by user some additional data

```
llistaDadesUsuari = serviceLocator.getUserService().findUserDataByUserName("test");
for (var i=0; i<llistaDadesUsuari.size(); i++) {
  var dadaUsuari = llistaDadesUsuari.get(i);
  out.println("Atributs " + dadaUsuari.attribute + " = " + dadaUsuari.value);
}
```

Recover users from a json query with AI

```
/** Print on screen the names of all users whose username contains the letter a
 **/
var userService = serviceLocator.getService("com.soffid.iam.base.service.UserService");
var query = new com.soffid.zkdb.api.Query();
query.setFilter('userName co "a"'); // SCIM filter for username containing 'a'

var pagedResult = userService.findUsers(query);
var users = pagedResult.getResources();

if (users && users.size() > 0) {
  out.println("Users whose username contains 'a':");
  for (var i = 0; i < users.size(); i++) {
    var user = users.get(i);
    out.println(user.userName);
  }
} else {
  out.println("No users found with 'a' in their username.");
}
```

Create a new identity

```
var newUser = new com.soffid.iam.base.api.User();

newUser.userName = "jkepler";
newUser.firstName = "Johannes";
newUser.lastName = "Kepler";
newUser.userType = "I";
newUser.primaryGroup = "world";
newUser.active = true;

serviceLocator.getUserService().create(newUser);
out.println("Created "+newUser.userName);
```

Update an identity

```
var  u = serviceLocator.getUserService().findUserByUserName("jkepler");
u.userType = "E";
u = serviceLocator.getUserService().update(u);
out.println("Updated "+u.userName);
```

Delete an identity

```
var u = serviceLocator.getUserService().findUserByUserName("jkepler");
if (u!=null) {
  serviceLocator.getUserService().delete(u);
  out.println("Deleted "+u.userName);
} else {
  out.println("User not found");
}
```

Account scripts

Recover accounts of users in Soffid 3

```
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");
```

```
}
```

Recover accounts of users in Soffid 4 with AI with pagination

```
/** search all account whose owner's userName contains the letter 'd' and print the name of the account and the system by the screen
 **/
var query = new com.soffid.zkdb.api.Query();
query.filter = "users.user.userName co \"a\"";
query.pageSize = 2;
query.startIndex = 0;

var pagedResult;
do {
    pagedResult = serviceLocator.getAccountService().findAccounts(query);
    var accounts = pagedResult.resources;

    for (var i = 0; i < accounts.size(); i++) {
        var account = accounts.get(i);
        out.println("Account: " + account.name + ", System: " + account.system);
    }

    query.startIndex += query.pageSize;
} while (query.startIndex < pagedResult.totalResults);
```

Remove attribute values of a metadata in Soffid 3

```
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","AD");
```

Remove attribute values of a metadata in Soffid 4

```
function removeUnAttributeValues(attribute, system) {
  var query = new com.soffid.zkdb.api.Query();
  query.filter = "system eq \"" + system + "\"";

  var pagedResult = serviceLocator.getAccountService().findAccounts(query);
  var la = pagedResult.getResources();

  for (var i = 0; i < la.size(); i++) {
    var a = la.get(i);
    var laa = serviceLocator.getAccountService().getAccountAttributes(a);
    for (var j = 0; j < laa.size(); j++) {
      var aa = laa.get(j);
      if (aa.attribute == attribute) {
        if (aa.value != null) {
          out.print("accountName: " + a.name + ", attribute.value: " + aa.value);
          serviceLocator.getAccountService().removeAccountAttribute(aa);
          out.println(" ---> removed");
        }
      }
    }
  }
}
removeUnAttributeValues("manager", "AD");
```

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

```
var queryUsuaris = new com.soffid.zkdb.api.Query();
queryUsuaris.filter = "userName eq \"david.gomez\"";
var pagedUsuaris = serviceLocator.getUserService().findUsers(queryUsuaris);
var llistaUsuaris = pagedUsuaris.getResources();

for (var i = 0; i < llistaUsuaris.size(); i++) {
  var usuari = llistaUsuaris.get(i);

  var queryComptes = new com.soffid.zkdb.api.Query();
  queryComptes.filter = "users.user.userName eq \"" + usuari.userName + "\"";
  var pagedComptes = serviceLocator.getAccountService().findAccounts(queryComptes);
  var llisstacuentas = pagedComptes.getResources();

  for (var j = 0; j < llisstacuentas.size(); j++) {
    var cuenta = llisstacuentas.get(j);
    out.print("   Cuenta : " + cuenta.name);

    var llistaRole = serviceLocator.getApplicationService().findRoleAccountByAccount(cuenta.id);
    for (var k = 0; k < llistaRole.size(); k++) {
      var role = llistaRole.get(k);
      out.print("      Role: " + role.roleName + "\n");
    }
  }
}
```

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

```
var queryUsuaris = new com.soffid.zkdb.api.Query();
queryUsuaris.filter = "userName eq \"david.gomez\"";
var pagedUsuaris = serviceLocator.getUserService().findUsers(queryUsuaris);
var llistaUsuaris = pagedUsuaris.getResources();

for (var i = 0; i < llistaUsuaris.size(); i++) {
  var usuari = llistaUsuaris.get(i);

  var queryComptes = new com.soffid.zkdb.api.Query();
  queryComptes.filter = "users.user.userName eq \"" + usuari.userName + "\"";
  var pagedComptes = serviceLocator.getAccountService().findAccounts(queryComptes);
  var llisstacuentas = pagedComptes.getResources();

  for (var j = 0; j < llisstacuentas.size(); j++) {
    var cuenta = llisstacuentas.get(j);
    out.print("   Cuenta : " + cuenta.name);
    out.println("   ID: " + cuenta.id);

    var llistaRole = serviceLocator.getApplicationService().findRoleAccountByAccount(cuenta.id);
    for (var k = 0; k < llistaRole.size(); k++) {
      var role = llistaRole.get(k);
      out.print("      Role: " + role.roleName + "\n");
      out.println("          Aplicacion: " + role.informationSystemName);
    }
  }
}
```

Print the roles associated with each account

```
var query = new com.soffid.zkdb.api.Query();
query.filter = "";
var paged = serviceLocator.getUserService().findUsers(query);
var usuCuenta = paged.getResources();

for (var i = 0; i < usuCuenta.size(); i++) {
  var listaUsuCuenta = usuCuenta.get(i);

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

```
  var rolsUser = serviceLocator.getUserService().findUserRolesHierachyByUserName(listaUsuCuenta.userName);
  for (var j = 0; j < rolsUser.size(); j++) {
    var listaRolsUser = rolsUser.get(j);
    out.println("Nombre del Rol: " + listaRolsUser.name);
    out.println("Descripcion: " + listaRolsUser.description);
    out.println();
  }
}
```

Create a new role

```
try {
  var newRol = new com.soffid.iam.iga.api.Role();
  newRol.name = "Rol_New_Script";
  newRol.description = "Rol Script";
  newRol.informationSystemName = "SOFFID";
  newRol.system = "soffid";
  serviceLocator.getApplicationService().create(newRol);
  out.println("Created: " + newRol.name);

} catch(e) {
    out.println("Error: " + e);
}
```

Update a role

```
var query = new com.soffid.zkdb.api.Query();
query.filter = "name eq \"Rol editado por script\" and informationSystemName eq \"APPLICATION01\"";

var pagedResult = serviceLocator.getApplicationService().findRoles(query);
var editRole = pagedResult.getResources();

for (var i = 0; i < editRole.size(); i++) {
    var role = editRole.get(i);
    out.println(role.name);
    role.name = "ROL01";
    try {
        role = serviceLocator.getApplicationService().update(role);
        out.println(role.name);
```

```
    } catch(e) {
        out.println("Error: " + e.message);
        out.println("Stack: " + e.stack);
    }
}
```

Delete a role

```
try {
  var editRole = serviceLocator.getApplicationService().findRoleById(16576);
  serviceLocator.getApplicationService().delete(editRole);
} catch(e) {
    out.println("Error: " + e.message);
}
```

List the roles of an application

```
var query = new com.soffid.zkdb.api.Query();
query.filter = "informationSystemName eq \"SOFFID\"";

var pagedResult = serviceLocator.getApplicationService().findRoles(query);
var list = pagedResult.getResources();

for (var i = 0; i < list.size(); i++) {
    var role = list.get(i);
    out.println(role.name);
}
```

Mail scripts

Send a simple email

```
serviceLocator.getMailService().sendTextMail("user@domian.com", "Test", "Hello world!");
out.println("Mail sent!");
```

Send emails with attached files

```
import javax.mail.BodyPart;
import javax.mail.internet.MimeBodyPart;
import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import java.util.ArrayList;
path = "/tmp/";
name = "file.txt";
BodyPart att = new MimeBodyPart();
att.setDataHandler(new DataHandler(new FileDataSource(path+name)));
att.setFileName(name);
to = "aretha@soffid.com";
cc = "etaylor@soffid.com";
subject = "This is an email with attachment ";
body = "In this email you can see an attachment.";
mimeBodyParts = new ArrayList();
mimeBodyParts.add(att);

serviceLocator.getMailService().sendHtmlMail(to, subject, body, mimeBodyParts);
serviceLocator.getMailService().sendHtmlMail(to, cc, subject, body, mimeBodyParts);
serviceLocator.getMailService().sendTextMailToActors(new String[]{"aretha"}, subject, body, mimeBodyParts);
serviceLocator.getMailService().sendTextMailToActors(new String[]{"aretha"}, cc, subject, body,
mimeBodyParts);
out.println("Mails sent!");
```

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);
```

```
  }
}
```

On user change

Run a Python script when the user has assigned an specific role

```
if (user != null) {
  roleGrantList = serviceLocator.getApplicationService().findEffectiveRoleGrantByUser(user.id);
  for(roleGrant:roleGrantList){
    if (roleGrant.roleName.equals("SOFFID_TEST")) {
      // RUN SCRIPT
      String command = "python3 /opt/soffid/iam-console-3/conf/exampleScript.py > /opt/soffid/iam-console-
3/conf/resultscript01.txt";
      Process process = Runtime.getRuntime().exec(command);
      user.comments = "ADD comments";
      user = serviceLocator.getUserService().update(user);
    }
  }
}
```

Agent scripts

User full name

```
return firstName + lastName;
```

Create mainDomain if it doesn't exit

```
var mailDomain = "exampledomain";
if (mailDomain != null && mailDomain.contains("@")) {
    var mailTokens = email.split("@");
    mailDomain = mailTokens[1];
}

var service = serviceLocator.getMailListsService();
var domain = service.findMailDomainByName(mailDomain);

if (domain == null) {
```

```
    domain = new com.soffid.iam.iga.api.MailDomain();  // ← iga.api
    domain.setCode(mailDomain);
    domain.setDescription(mailDomain);
    domain.setObsolete(new java.lang.Boolean(false));
    domain = service.create(domain);
}

return mailDomain;
```

Recover active agents

```
var llistaAgents = serviceLocator.getDispatcherService().findAllActiveDispatchers();
for (var i = 0; i < llistaAgents.size(); i++) {
    var agent = llistaAgents.get(i);
    out.println("Nom: " + agent.name);
    out.println("Class Name: " + agent.className + "\n");
}
```

Show by a user the agents that have associates

```
var queryUsuaris = new com.soffid.zkdb.api.Query();
queryUsuaris.filter = "userName eq \"admin\"";
var pagedUsuaris = serviceLocator.getUserService().findUsers(queryUsuaris);
var llistaUsuaris = pagedUsuaris.getResources();

for (var i = 0; i < llistaUsuaris.size(); i++) {
    var usuari = llistaUsuaris.get(i);
    out.println("Usuario: " + usuari.userName);

    var queryComptes = new com.soffid.zkdb.api.Query();
    queryComptes.filter = "users.user.userName eq \"" + usuari.userName + "\"";
    var pagedComptes = serviceLocator.getAccountService().findAccounts(queryComptes);
    var llisstacuentas = pagedComptes.getResources();

    for (var j = 0; j < llisstacuentas.size(); j++) {
        var cuenta = llisstacuentas.get(j);
        out.print("   Cuenta : " + cuenta.name);
        out.println("   ID: " + cuenta.id);

        var llistaRole = serviceLocator.getApplicationService().findRoleAccountByAccount(cuenta.id);
```

```
        for (var k = 0; k < llistaRole.size(); k++) {
            var role = llistaRole.get(k);
            out.print("      Role: " + role.roleName + "\n");
            out.println("          Aplicacion: " + role.informationSystemName);
            out.println("            Agente: " + role.system);
        }
    }
}
```

Revision #26
Created 19 July 2025 11:57:16 by Sion Vives
Updated 4 March 2026 16:01:31 by David Gómez