

Further information

Further information

- [My Profile](#)
- [Soffid Objects \(for agent mappings\)](#)
- [Sample scripts](#)
- [Utility classes](#)
- [Beanshell vs Javascript](#)
- [Office 365 as External SAML identity provider](#)

My Profile

Description

My Profile is a part of the Identity self service that allows end users to configure their own profile, update the user info and preferences, change their password, and recover questions.


To view My Profile, you must select the My Profile option that will be displayed when you click on the drop-down menu at the top right. Then Soffid displays a new window that will allow end users to configure their profiles.

Screen overview


The screenshot displays the Soffid user interface for the 'My Profile' section. On the left is a navigation sidebar with the Soffid logo and menu items: IBS, IGA, AM, PAM, IBC, and ADV. The main content area has a search bar and tabs for 'Basics', 'Authorizations', and 'Application consents'. The 'Basics' tab is active, showing 'User Info' and 'Preferences' sections. The 'User Info' section includes 'Last login: 08/08/2025 10:40' and 'Last IP connection: 172.19.0.1', with a 'Change Password' button. The 'Preferences' section includes dropdowns for 'Language' (English), 'Time zone' (Central European Time (GMT+2)), and text inputs for 'Date format' (dd/MM/yyyy), 'Sample' (08/08/2025), and 'Time format' (HH:mm).

New Password


Current password

New password

New password

Password policy

```
!PasswordServiceImpl.PasswordMaxDurationCondition!
```

Related objects

- Users : to display the roles granted to a user
- Roles : to display the roles
- Information systems : to display the roles through the information systems
- Authorizations : to review the authorizations and manage the roles assigned

Standard attributes

Basic

User Info

- **Last login:** date and time of the user's last login.
- **Last IP connection:** IP of the user's last login.
- **Change password:** allows end-users to change their password.
- **Password recovery questions:** (only when add-on retrieve passwords is configured) allows end-users to config their own questions to recover their passwords.

For more info about password recovery, you can visit the [Password recovery questions page](#).

Preferences

- **Language:** allows end-users to select their preferred language.
- **Time zone:** allows end-users to select their time zone.
- **Date format:** allows end-users to select the format date.
- **Sample:** displays how the date will be displayed in Soffid Console
- **Time format:** allows end-users to select the format time
- **Sample:** displays how the time will be displayed in Soffid Console
- **Enable desktop notifications in this browser:** enable desktop notifications in this browser
- **Display:** Light (background in white), dark (background in dark)

Authorizations

Display a list with the user authorizations.

- **Role:** role granted
- **Authorization [domain value]:** authorization description
- **ITS Scope:** authorization scope
- **Domain value:** domain where the role granted is assigned (* when there is no domain)

Application consents

Displays a list of all the user's consents given, and the user can see all of them. Users can remove the consent at any time as well.

When the user connects to a new application, the IdP will indicate which data will be shared with this application. That information is defined in the Attribute sharing policies page of the Federation.

For more info about password recovery, you can visit the [Attribute sharing policies page](#).

Actions

Change password	<p>Allows the user to change their current password.</p> <p>The pop-up will display the restrictions applied according to your password policy.</p> <p>You must enter your current password. If you cannot remember it, it is best to use the password recovery option when logging in to the Console. This option is included in the password recovery add-on.</p>
Undo	<p>Allows you to undo any changes made.</p>
Apply changes	<p>Allows you to save the data. Once you apply changes, the details page will be closed.</p>

Soffid Objects (for agent mappings)

You can consult the list of Soffid attributes:

1. [User Object](#)
 2. [Account Object](#)
 3. [Group Object](#)
 4. [Role Object](#)
 5. [Grant Object](#)
 6. [Maillist Object](#)
 7. [Membership Object](#)
 8. [dispatcherService](#)
 9. [Authoritative change object](#)
-

User object

A user objects are maps that hold the information belonging to a single user account.

Attribute	Type	Description
id	Long	user id
accountId	Long	account id
accountName	String	account name
system	String	managed system (agent) name
accountDescription	String	account description

Attribute	Type	Description
active	Boolean	true if user is active
accountDisabled	Boolean	true if account is disabled
mailAlias	String	blank separated mails
userName	String	user name
primaryGroup	String	user's primary group name
comments	String	user's comments
createdOn	Date	user creation date
modifiedOn	Date	user last modification date
mailDomain	Date	user mail domain (email right side of @)
fullName	String	user full name
shortName	String	user mail name (email left side of @)
firstName	String	user first name
lastName	String	user last name
lastName2	String	user second last name (when applicable)
mailServer	String	mail server host name
homeServer	String	home drive server host name
profileServer	String	roaming profile server host name
phone	String	user's phone number
userType	String	user type
createdBy	String	user name creator of this user
modifiedBy	String	user name modifier of this user

Attribute	Type	Description
secondaryGroups	List<Map<String,Object>>	list of groups the user belongs to, including primary group The attributes of the inner map are described later
attributes	Map<String,String>	additional user attributes
grantedRoles	List<Map<String,Object>>	list of grants directly granted to the user
allGrantedRoles	List<Map<String,Object>>	list of grants directly on indirectly granted to the user
granted	List<String>	list of role names and group names directly granted to the user
allGranted	List<String>	list of role names and group names directly or indirectly granted to the user

Account object

An account object holds the information belonging to an account.

Attribute	Type	Description
accountDescription	String	account description
accountDisabled	Boolean	true if account is disabled
accountId	Long	account id
accountName	String	account name
allGranted	List<String>	list of role names directly or indirectly granted to the user
allGrantedRoles	List<Map<String,Object>>	list of grants directly on indirectly granted to the user
attributes	Map<String,String>	additional account attributes
granted	List<String>	list of role names directly granted to the user
grantedRoles	List<Map<String,Object>>	list of grants directly granted to the user
lastLogin	Calendar	lastLogin

Attribute	Type	Description
lastPasswordUpdate	Calendar	lastPasswordUpdate
lastUpdate	Calendar	lastUpdate
passwordExpiration	Calendar	passwordExpiration
passwordPolicy	String	password policy
system	String	managed system (agent) name
type	AccountType	"U"=user, "S"=shared, "P"=privileged, "I"=ignored

Group object

An group object holds the information belonging to a group.

Attribute	Type	Description
groupId	Long	group id
name	String	group name
description	String	group description
parent	String	parent group name
server	String	home server host name
disabled	boolean	true if the group is disabled
accountingGroup	String	group accounting information
type	String	group type
driveLetter	String	home server letter to connect to
users	List<Map<String,Object>>	list of <u>users</u> belonging to this group
userNames	List<String>	list of user names belonging to this group
allUsers	List<Map<String,Object>>	list of <u>users</u> directly or indirectly belonging to this group

Attribute	Type	Description
allUserNames	List<String>	list of user names either directly or indirectly grantee of this role
grantedRoles	List<Map<String,Object>>	list of <u>roles</u> granted to this group
grantedRoleNames	List<String>	list of role names granted to this group

Role object

An role object holds the information belonging to a role.

Attribute	Type	Description
roleId	Long	role id
system	String	managed system (agent) name
name	String	role name
application	String	application system name
category	String	role category
passwordProtected	boolean	true if role should be password protected (where applicable)
description	String	Role description
wfmanaged	boolean	true if role should be displayed in self service requests
domain	String	custom domain for this role: Use com.soffid.iam.api.DomainType constants or configured custom domain
ownedRoles	List<Map<String,Object>>	list of <u>roles granted</u> to this one
ownerRoles	List<Map<String,Object>>	list of <u>roles grantee</u> of this one
ownerGroups	List<Map<String,Object>>	list of <u>groups</u> grantee of this role
grantedAccountNames	List<String>	list of account names directly grantee of this role
grantedAccounts	List<Map<String,Object>>	list of <u>users</u> directly grantee of this role

Attribute	Type	Description
allGrantedAccountNames	List<String>	list of account names either directly or indirectly grantee of this role
allGrantedAccounts	List<Map<String,Object>>	list of users either directly or indirectly grantee of this role
attributes	Map<String,Object>	role's custom attributes

Grant object

Grant, grantedRole & allGrantedRoles

The objects grant, grantedRole and allGrantedRoles are used to assing roles to accounts and roles.

Attribute	Type	Description
domainValue	String	grant value (if any)
grantedRole	String	granted role name
grantedRoleId	Long	granted role id
grantedRoleObject	<u>role object</u>	granted role
grantedRoleSystem	String	granted role managed system (agent) name
id	Long	grant id
ownerAccount	String	grantee account name
ownerAccountObject	<u>account object</u>	grantee account
ownerGroup	String	grantee group name
ownerRoleId	String	grantee role id
ownerRoleName	String	grantee role name
ownerSystem	String	grantee account or role managed system name
ownerUser	String	grantee user name

Examples

Grant

Example to map a grant object (assign a role to an account):

System attribute	Direction	Soffid attribute
role_name	=>	grantedRole
account_name	=>	ownerAccount

GrantedRole

Example to map a grantedRole object (assign a role as a child of another role):

System attribute	Direction	Soffid attribute
role_name	=>	grantedRole
parent_role_name	=>	ownerRoleName

AllGrantedRoles

Example to map a allGrantedRoles object in a holderGroup (assign a role to an account in a specific group):

System attribute	Direction	Soffid attribute
role_name	=>	grantedRole
parent_role_name	=>	ownerRoleName
group_code	=>	domainValue
group_code	=>	holderGroup
userName	=>	ownerUser

Maillist object

Attribute	Type	Description
id	Long	internal mail list id
name	String	mail list name (the initial part, before the @ sign)
domain	String	mail list domain (the remaining part after the @ sign)

Attribute	Type	Description
system	String	managed system (agent) name
description	String	mail list description
users	String array	user names that are bound to this mail list
groups	String array	group names thta are subscribed to this mai list
roles	String array	role names that grant access to this mail list
lists	String array	Nested mail lists
explodedUsers	String array	Names of the users that should be subscribed to this mail list, including the users that should be subscribed due to group or role membership
explodedUserAddresses	String array	Mail addresses of any exploded User

Membership object

A membership object contains the user account information as well as the group the user belongs to.

Attribute	Type	Description
userName	String	User name
user	Map<String,Object>	<u>user object</u>
groupName	String	Group name
group	Map<String,Object>	<u>group object</u>
attributes	Map<String,Object>	Membership custom attributes

dispatcherService

dispatcherService is an object available from agents' attribute translation rules.

This object contains four methods:

method name	parameters	result type	comments
-------------	------------	-------------	----------

soffidToSystem	<u>ExtensibleObject</u> soffidObject	<u>ExtensibleObject</u>	Uses attribute translation tables to transform a soffid object to a target system object. Mind to fill-in objectType property to use the proper object mapping
systemToSoffid	<u>ExtensibleObject</u> systemObject	<u>ExtensibleObject</u>	Uses attribute translation tables to transform a target system object to a Soffid object. Mind to fill-in objectType property to use the proper object mapping
search	<u>ExtensibleObject</u> exampleObject	<u>ExtensibleObject</u>	Uses the exampleObject to perform a query by example on the target system. If the object exists on the target system, it is returned. Mind to fill-in objectType property with the desired system object type
invoke	String verb String action Map parameters	List of Map	This method allows arbitrary executions on the target system, but its semantics can change depending on the connector used. For instance, it can be used to perform a GET on the target system in REST connector, can issue an LDAP query on ActiveDirectory connector, can execute a SELECT sentence on a SQL connector, or can execute an operating system command in Shell connector. The results are returned as a list of objects (map).

Examples

Snippet to query the sys_id attribute for a grant owner

```
System.out.println("Searching id for "+ownerRoleName);
com.soffid.iam.sync.intf.ExtensibleObject eo = new com.soffid.iam.sync.intf.ExtensibleObject();
```

```

eo.setObjectType("ROLE");
eo{"name"} = ownerRoleName;
eo = dispatcherService.search(eo);
System.out.println("FOUND "+eo{"sys_id"});
return eo{"sys_id"};

```

Snippet that performs a REST query to get group to role assignments in ServiceNow

```

list = dispatcherService.invoke ("GET",
    "https://arxusdev.service-
now.com/api/now/table/sys_group_has_role?sysparm_exclude_reference_link=true&sysparm_display_value
=all&sysparm_fields=role%2Cgroup&sysparm_query=group="+sys_id,
    null).
    get(0).get("result")

r = new java.util.LinkedList();
for ( d: list)
{
    grant = new java.util.HashMap();
    grant{"grantedRole"} = d.get("role").get("display_value");
    grant{"grantedRoleSystem"} = "ServiceNow";
    grant{"ownerRoleName"} = name;
    grant{"ownerSystem"} = "ServiceNow";
    r.add (grant);
}
return r;

```

Snippet of invoke usage on a relational database

```

// Table ITREPRT
role = source{"granted"}.size() == 0 ? "" : source{"granted"}.get(0);
System.out.println ("***** ROLE "+role);
args = new java.util.HashMap();
args.put("user", source{"accountName"}.toUpperCase());
if (role.equals ("Receptores PR") || role.equals("Jefes_Personal")) {
    r = dispatcherService.invoke("select", "* from ITREPRT where IDUSER=:user", args);
    if (r.size() == 0) {
        dispatcherService.invoke("insert", "into ITREPRT(IDUSER,NOME) values (:user, 1)", args);
    }
}

```

```

} else {
    dispatcherService.invoke("delete", "from ITREPRT where IDUSER=:user", args);
}
// TABLE MRGEUCT
cc = source{"attributes"}{"dominio"};
if ( source{"userType"} .equals ("T")) {
    cc = source{"userName"}.substring(1);
}
while (cc != null && cc.startsWith("0")) cc = cc.substring(1);
System.out.println ("***** COST CENTER "+cc);
if (cc != null && ! cc.trim().isEmpty())
{
    args = new java.util.HashMap();
    args.put("user", source{"accountName"}.toUpperCase());
    args.put("cc", cc);
    r = dispatcherService.invoke("SELECT", "* from MRGEUCT where IDUSER=:user and MOARPR=:cc", args);
    if (r.size() == 0) {
        dispatcherService.invoke("INSERT", "into MRGEUCT(MOARPR,CENTRA, IDUSER, NOTIFI ) "+
            "values ('I', :cc, :user, 'S')", args);
        dispatcherService.invoke("INSERT", "into MRGEUCT(MOARPR,CENTRA, IDUSER, NOTIFI ) "+
            "values ('BM', :cc, :user, 'S')", args);
        dispatcherService.invoke("DELETE", "FROM MRGEUCT WHERE CENTRA!:=:cc AND IDUSER=:user", args);
    }
}
return true;

```

Snippet of invoke usage on a Active Directory I

```

hashMap = new java.util.HashMap();
list = serviceLocator.getDispatcherService().invoke("AD soffid.pat",
    "select",
    "(&(objectClass=user))",
    hashMap);
out.println("** list.size -- " + list.size());

```

Snippet of invoke usage on a Active Directory II

```

ACC = source{"accountName"};
la = dispatcherService.invoke("AD soffid.pat", "(&(objectClass=user)(sAMAccountName=userName))", new

```

```
java.util.HashMap();
```

Authoritative change object

A user objects are maps that hold the information belonging to a single user account

Attribute	Type	Description
id	Long	user id
accountId	Long	account id
accountName	String	account name
system	String	managed system (agent) name
accountDescription	String	account description
active	Boolean	true if user is active
accountDisabled	Boolean	true if account is disabled
mailAlias	String	blank separated mails
userName	String	user name
primaryGroup	String	user's primary group name
comments	String	user's comments
createdOn	Date	user creation date
modifiedOn	Date	user last modification date
mailDomain	Date	user mail domain (email right side of @)
fullName	String	user full name
shortName	String	user mail name (email left side of @)
firstName	String	user first name
lastName	String	user last name

Attribute	Type	Description
lastName2	String	user second last name (when applicable)
mailServer	String	mail server host name
homeServer	String	home drive server host name
profileServer	String	roaming profile server host name
phone	String	user's phone number
userType	String	user type
createdBy	String	user name creator of this user
modifiedBy	String	user name modifier of this user
secondaryGroups	List<Map<String,Object>>	list of groups the user belongs to, including primary group The attributes of the inner map are described in the link
secondariGroups2	List<Map<String,Object>>	list of user memberships , excluding primary group The attributes of the inner map are described link
attributes	Map<String,String>	additional user attributes
grantedRoles	List<Map<String,Object>>	list of grants directly granted to the user
allGrantedRoles	List<Map<String,Object>>	list of grants directly on indirectly granted to the user
granted	List<String>	list of role names and group names directly granted to the user
allGranted	List<String>	list of role names and group names directly or indirectly granted to the user

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/resultsript01.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 llistacuentas = pagedComptes.getResources();

    for (var j = 0; j < llistacuentas.size(); j++) {
        var cuenta = llistacuentas.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);  
    }  
}  
}
```

Utility classes

Crypt

Crypt allows to encrypt text with different algorithms and verify the resulting hash.

To use this class: `com.soffid.iam.crypt.Crypt`

All methods are static:

```
hash(String algorithm, String text) -> String
pBKDF2Sha256(String text, String utf8Salt, int iterations) -> String
pBKDF2Sha256(String text, byte []salt, int iterations) -> String
pBKDF2Sha1(String text, String utf8Salt, int iterations) -> String
pBKDF2Sha1(String text, byte []salt, int iterations) -> String
genSaltBytes() -> byte[] // 8 bytes
genSaltBytes(int size) -> byte[]
genSalt() -> String // 8 bytes
genSalt(int size) -> String
verify(String algorithm, String text, String hash) -> boolean
```

The algorithms allowed are:

- bcrypt
- pBKDF2Sha256
- pBKDF2Sha1 (or pBKDF2)
- Base64 (used by default is the algorithm is not in the previous list)

One example:

```
String myText = "abcd";
String myAlgorithm = "bcrypt";
String myHash = com.soffid.iam.crypt.Crypt.hash(myAlgorithm, myText);
boolean isVerified = com.soffid.iam.crypt.Crypt.verify(myAlgorithm, myText, myHash);
if (isVerified) {
    return myHash;
} else {
```

```
return null;  
}
```

CalendarConverter

CalendarConverter allows to covert Calendar into String.

To use this class: `com.soffid.iam.json.CalendarConverter`

The methods (non static):

```
toString(Calendar instance) -> String  
fromString(final String text) -> Calendar
```

One example:

```
out.println(new com.soffid.iam.json.CalendarConverter().toString(date));
```

Beanshell vs Javascript

Description

Soffid 4 configures the smart engine with **Javascript** scripting language as the default. See [Smart engine settings](#).

Previously, the default engine was Beanshell, and many scripts will need to be adapted.

This page lists these differences.

Related objects

- [Smart engine settings](#) : where the engine is configured.
- Where we can use scripts:
 - [Agents](#) : properties, mappings and triggers
 - [Custom scripts](#) : all the scripts
 - [Account naming rules](#) : script to validate and set name
 - [Role assignment rules](#) : script to validate
 - [BPM editor](#) : visualization, triggers, transitions
 - [Password policies](#) : optional script

Table of differences

Topic	Beanshell	Javascript
-------	-----------	------------

variable	s = "text";	// The use of var should be mandatory, //but it almost always works without using it. var s = "text"; or s = "text";
function	public void doSomething(String system) { ... } doSomething("APP_USERS");	function doSomething(system) { ... } doSomething("APP_USERS");
for	for (user : listOfUsers) { ... }	for (var i=0; i<listOgUsers.size(); i++) { ... }
equals	user == null	user === null
equals	userName.equals("myName")	userName === "myName"
java class	es.caib.seycon.ng.comu.AccountType	Java.type("es.caib.seycon.ng.comu.AccountType") // We can also generate objects without Java.type query = new com.soffid.zkdb.api.Query(); or Query = Java.type("com.soffid.zkdb.api.Query"); query = new Query();
catch / printStackTrace	} catch(Exception e) { e.printStackTrace(out); }	} catch(e) { out.println(e.message); }
	e.printStackTrace(out)	out.println(e.message) + out.println(e.stack)

Search in Soffid 4

In Soffid 4 the findObjectByJsonQuery method no longer exists, it has been replaced by findObjects with or without pagination.

List users

```
q = new com.soffid.zkdb.api.Query();
pr = serviceLocator.getUserService().findUsers(q);
lu = pr.getResources();
for (i=0; i<lu.size(); i++) {
    u = lu.get(i);
    out.println(u.userName);
}
```

List users with pagination

```
us = serviceLocator.getUserService();
q = new com.soffid.zkdb.api.Query();
q.startIndex = 0;
q.pageSize = 2;
do {
    out.println("Searching...");
    pr = us.findUsers(q);
    lu = pr.getResources();
    for (i=0; i<lu.size(); i++) {
        u = lu.get(i);
        out.println(" "+u.userName);
    }
    q.startIndex = q.startIndex+pr.itemsPerPage;
} while (q.startIndex<pr.totalResults);
```

Office 365 as External SAML identity provider

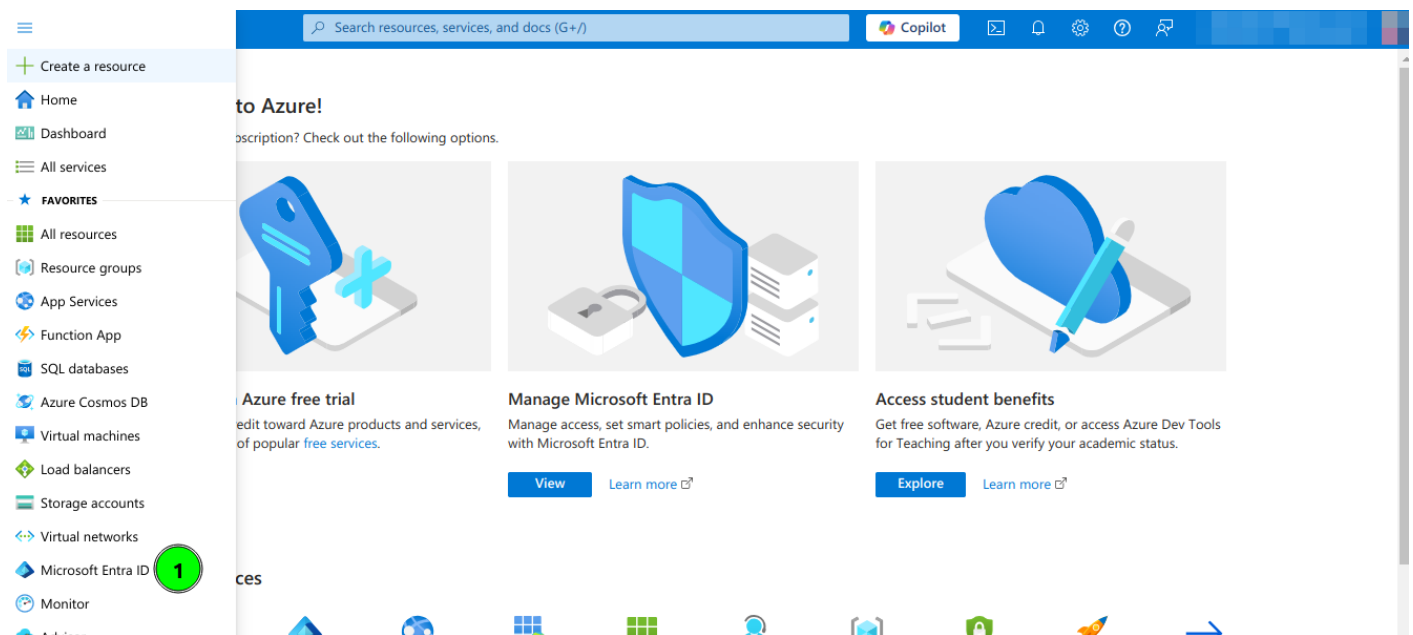
Introduction

Steps to configure Office 365 as External SAML identity provider.

Step-by-Step

“1. Open a <https://portal.azure.com>

2. Open **Microsoft Entra ID** and then select **Enterprise applications** option



Microsoft Azure Search resources, services, and docs (G+)

Home > Soffid IAM, S.L. | Overview

Overview | Monitoring | Properties | Recommendations | Setup guides

Search your tenant

Basic information

Name	Soffid IAM, S.L.	Users	10
Tenant ID	25bc9848-d3c2-423f-93f1-cf465332a68d	Groups	5
Primary domain	soffidcom.onmicrosoft.com	Applications	10
License	Microsoft Entra ID Free	Devices	1

Alerts

Migrate to the converged Authentication methods policy
Please migrate your authentication methods off the legacy MFA and SSPR policies by September 2025 to avoid any service impact

Enterprise applications

3. Select All applications and click New Application

Microsoft Azure Search resources, services, and docs (G+)

Home > Enterprise applications

Enterprise applications | All applications

New application Refresh Download (Export) Preview info Columns Preview features Got feedback?

View, filter, and search applications in your organization that are set up to use your Microsoft Entra tenant as their Identity Provider.

The list of applications that are maintained by your organization are in [application registrations](#).

Search by application name or object ID Application type == Enterprise Applications Application ID starts with Add filters

11 applications found

Name	Object ID	Application ID	Homepage URL	Created on	Certificate Expir...	Active Certificat.
GE				/12/2020	-	-
S				0/23/2024	Current	10/23/2027
SC				/20/2020	-	-
O				/25/2022	-	-
				0/23/2024	-	-
PA				/29/2020	-	-
TS				1/28/2023	-	-
AI				/10/2020	-	-

All applications

4. Select Create your own application

Browse Microsoft Entra Gallery



+ Create your own application | Got feedback?


The Microsoft Entra App Gallery is a catalog of thousands of apps that make it easy to deploy and configure single sign-on (SSO) and automated user provisioning. When deploying an app from the App Gallery, you leverage prebuilt templates to connect your users more securely to their apps. Browse or create your own application here. If you are wanting to publish an application you have developed into the Microsoft Entra Gallery for other organizations to discover and use, you can file a request using the process described in [this article](#).

Search application


Single Sign-on : All User Account Management : All Categories : All

Cloud platforms


Amazon Web Services (AWS)



Google Cloud Platform



Oracle




SAP

5. Type the name of your app and select the "Integrate any other application you don't find in the gallery (Non-gallery)" option

Create your own application



 Got feedback?

If you are developing your own application, using Application Proxy, or want to integrate an application that is not in the gallery, you can create your own application here.

What's the name of your app?

pat.soffid.lab 

What are you looking to do with your application?

- Configure Application Proxy for secure remote access to an on-premises application
- Register an application to integrate with Microsoft Entra ID (App you're developing)
- Integrate any other application you don't find in the gallery (Non-gallery)

[Create](#)

6. Click on **Set up single sign on**

Microsoft Azure Search resources, services, and docs (G+)

Home > Enterprise applications | All applications > Browse Microsoft Entra Gallery >

pat.soffid.lab | Overview

Enterprise Application

Overview

- Deployment Plan
- Diagnose and solve problems
- Manage
 - Properties
 - Owners
 - Roles and administrators
 - Users and groups
 - Single sign-on
 - Provisioning
 - Application proxy
 - Self-service
 - Custom security attributes
- Security
- Activity

Properties

Name: pat.soffid.lab

Application ID: 6bb554ed-b96d-4631-9204-...

Object ID: 76afac14-c10f-4dde-94f7-8d...

Getting Started

1. Assign users and groups
Provide specific users and groups access to the applications
[Assign users and groups](#)
2. Set up single sign on
Enable users to sign into their application using their Microsoft Entra credentials
[Get started](#)
3. Provision User Accounts
4. Conditional Access

7. Click the **SAML** option

Microsoft Azure Search resources, services, and docs (G+)

Home > Enterprise applications | All applications > Browse Microsoft Entra Gallery > pat.soffid.lab

pat.soffid.lab | Single sign-on

Enterprise Application

Overview

- Deployment Plan
- Diagnose and solve problems
- Manage
 - Properties
 - Owners
 - Roles and administrators
 - Users and groups
 - Single sign-on
 - Provisioning
 - Application proxy
 - Self-service
 - Custom security attributes
- Security
- Activity

Single sign-on (SSO) adds security and convenience when users sign on to applications in Microsoft Entra ID by enabling a user in your organization to sign in to every application they use with only one account. Once the user logs into an application, that credential is used for all the other applications they need access to. [Learn more.](#)

Select a single sign-on method [Help me decide](#)

- Disabled
Single sign-on is not enabled. The user won't be able to launch the app from My Apps.
- 1. SAML
Rich and secure authentication to applications using the SAML (Security Assertion Markup Language) protocol.
- Password-based
Password storage and replay using a web browser extension or mobile app.
- Linked
Link to an application in My Apps and/or Office 365 application launcher.

8. Enter the **Basic SAML Configuration** and Save:

- **Identifier:** https://<YOUR-SERVER>/soffid-iam-console

- **Reply URL:** https://<YOUR-SERVER>/soffid/saml/log/post
- **Sign on URL:** https://<YOUR-SERVER>/soffid/
- **Logout URL:** https://<YOUR-SERVER>/soffid/saml/slo/post

Microsoft Azure Search resources, services, and docs (G+)

Home > Enterprise applications | All applications > soffid.pat.lab

soffid.pat.lab | SAML-based Sign-on

Enterprise Application

Upload metadata file | Change single sign-on mode | Test this application | Got feedback?

- Overview
- Deployment Plan
- Diagnose and solve problems
- Manage
 - Properties
 - Owners
 - Roles and administrators
 - Users and groups
 - Single sign-on**
 - Provisioning
 - Application proxy
 - Self-service
 - Custom security attributes

Set up Single Sign-On with SAML

An SSO implementation based on federation protocols improves security, reliability, and end user experiences and is easier to implement. Choose SAML single sign-on whenever possible for existing applications that do not use OpenID Connect or OAuth. [Learn more.](#)

Read the [configuration guide](#) for help integrating soffid.pat.lab.

1 Basic SAML Configuration Edit

Identifier (Entity ID)	https://pat.soffid.lab:8443/soffid-iam-console
Reply URL (Assertion Consumer Service URL)	https://pat.soffid.lab:8443/soffid/saml/log/post
Sign on URL	https://pat.soffid.lab:8443/soffid/
Relay State (Optional)	Optional
Logout Url (Optional)	https://pat.soffid.lab:8443/soffid/saml/slo/post

2 Attributes & Claims Edit

givenname	user.givenname
-----------	----------------

Basic SAML Configuration



Save | Got feedback?

Identifier (Entity ID) * ⓘ

The unique ID that identifies your application to Microsoft Entra ID. This value must be unique across all applications in your Microsoft Entra tenant. The default identifier will be the audience of the SAML response for IDP-initiated SSO.

Default

<input type="text" value="https://pat.soffid.lab:8443/soffid-iam-console"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="ⓘ"/>	<input alt="trash icon" type="image"/>
---	-------------------------------------	--------------------------	--------------------------------	--

[Add identifier](#)

Reply URL (Assertion Consumer Service URL) * ⓘ

The reply URL is where the application expects to receive the authentication token. This is also referred to as the "Assertion Consumer Service" (ACS) in SAML.

Index Default

<input type="text" value="https://pat.soffid.lab:8443/soffid/saml/log/post"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text" value="ⓘ"/>	<input alt="trash icon" type="image"/>
---	--------------------------	-------------------------------------	--------------------------------	--

[Add reply URL](#)

Sign on URL (Optional)

Sign on URL is used if you would like to perform service provider-initiated single sign-on. This value is the sign-in page URL for your application. This field is unnecessary if you want to perform identity provider-initiated single sign-on.

<input type="text" value="https://pat.soffid.lab:8443/soffid/"/>	<input checked="" type="checkbox"/>
--	-------------------------------------

Relay State (Optional) ⓘ

The Relay State instructs the application where to redirect users after authentication is completed, and the value is typically a URI or URI path that takes users to a specific location within the application

9. Configure **Attributes & Claims** and change the attributes and claims to send the mailnickname as the user identifier (nameid)

soffid.pat.lab | SAML-based Sign-on

Enterprise Application

Upload metadata file Change single sign-on mode Test this application Got feedback?

- Overview
- Deployment Plan
- Diagnose and solve problems
- Manage
 - Properties
 - Owners
 - Roles and administrators
 - Users and groups
 - Single sign-on**
 - Provisioning
 - Application proxy
 - Self-service
 - Custom security attributes
- Security
- Activity
- Troubleshooting + Support

Set up Single Sign-On with SAML

An SSO implementation based on federation protocols improves security, reliability, and end user experiences and is easier to implement. Choose SAML single sign-on whenever possible for existing applications that do not use OpenID Connect or OAuth. [Learn more.](#)

Read the [configuration guide](#) for help integrating soffid.pat.lab.

- 1 Basic SAML Configuration** Edit

Identifier (Entity ID)	https://pat.soffid.lab:8443/soffid-iam-console
Reply URL (Assertion Consumer Service URL)	https://pat.soffid.lab:8443/soffid/saml/log/post
Sign on URL	https://pat.soffid.lab:8443/soffid/
Relay State (Optional)	Optional
Logout Url (Optional)	https://pat.soffid.lab:8443/soffid/saml/slo/post
- 2 Attributes & Claims** Edit

givenname	user.givenname
surname	user.surname
emailaddress	user.mail
name	user.userprincipalname
Unique User Identifier	user.mailnickname
- 3 SAML Certificates**

Attributes & Claims

+ Add new claim + Add a group claim Columns | Got feedback?

Required claim

Claim name	Type	Value
Unique User Identifier (Name ID)	SAML	1 user.mailnickname [nam... ***

Additional claims

Claim name	Type	Value
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailadd...	SAML	user.mail ***
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	SAML	user.givenname ***
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	SAML	user.userprincipalname ***
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	SAML	user.surname ***

Advanced settings

10. Copy the App Federation Metadata Url

Microsoft Azure Search resources, services, and docs (G+) Copilot

Home > soffid.pat.lab

soffid.pat.lab | SAML-based Sign-on

Enterprise Application

Upload metadata file Change single sign-on mode Test this application Got feedback?

- Overview
- Deployment Plan
- Diagnose and solve problems
- Manage
 - Properties
 - Owners
 - Roles and administrators
 - Users and groups
 - Single sign-on**
 - Provisioning
 - Application proxy
 - Self-service
 - Custom security attributes
- Security
- Activity
- Troubleshooting + Support

givenname	user.givenname
surname	user.surname
emailaddress	user.mail
name	user.userprincipalname
Unique User Identifier	user.mailnickname

3 SAML Certificates

Token signing certificate Edit

Status: Active

Thumbprint: 560B064826325CB89FOCE229BDBDEE3A20E64587

Expiration: 10/23/2027, 4:14:18 PM

Notification Email: admin@soffidcom.onmicrosoft.com

App Federation Metadata Url: <https://login.microsoftonline.com/25bc9848-d3c2-423f-93f1-cf465332a68d/> **1**

Certificate (Base64): [Download](#)

Certificate (Raw): [Download](#)

Federation Metadata XML: [Download](#)

Verification certificates (optional) Edit

Required	No
Active	0
Expired	0

11. Configure the External SAML identity Provider in the Soffid Console Authentication page

soffid Search

Main Menu > Administration > Configuration > Security settings > Authentication **1**

Global status

No Maintenance mode (only administrators can log in)

Message to display before logging in:

Session timeout in minutes:

Username and password

Yes Enabled

No Forward authentication requests to trusted target systems

External SAML identity provider

Yes Enabled **2**

Soffid server host name:

SAML federation metadata URL: **3**

Cache limit (seconds):

Identity provider: **4**

SAML attribute containing user name:

No Enable SAML debug log

12. Optional, enable any user to login

soffid.pat.lab | Properties

Enterprise Application



Save Discard Delete Got feedback?

- Overview
- Deployment Plan
- Diagnose and solve problems
- Manage
 - Properties 1**
 - Owners
 - Roles and administrators
 - Users and groups
 - Single sign-on
 - Provisioning
 - Application proxy
 - Self-service
 - Custom security attributes
- Security
- Activity
- Troubleshooting + Support

View and manage application settings for your organization. Editing properties like display information, user sign-in settings, and user visibility settings requires Global Administrator, Cloud Application Administrator, Application Administrator roles. [Learn more.](#)

If this application resides in your tenant, you can manage additional properties on the [application registration](#).

Enabled for users to sign-in?	<input checked="" type="radio"/> Yes <input type="radio"/> No
Name *	<input type="text" value="soffid.pat.lab"/>
Homepage URL	<input type="text" value="https://account.activedirectory.windowsazure.com:444/applications/de..."/>
Logo	<div style="border: 1px solid #ccc; background-color: #0070c0; color: white; text-align: center; width: 40px; height: 40px; margin: 0 auto;">S</div> <input type="text" value="Select a file"/>
User access URL	<input type="text" value="https://launcher.myapps.microsoft.com/api/signin/b17d8844-4fac-4f2..."/>
Application ID	<input type="text" value="b17d8844-4fac-4f2c-863a-59d8be7781f5"/>
Object ID	<input type="text" value="38283f52-d45b-4306-bef5-1ea73fc7e7ef"/>
Terms of Service Url	<input type="text" value="Publisher did not provide this information"/>
Privacy Statement Url	<input type="text" value="Publisher did not provide this information"/>
Reply URL	<input type="text" value="https://pat.soffid.lab:8443/soffid/saml/log/post"/>
Assignment required?	<input type="radio"/> Yes <input checked="" type="radio"/> No 2
Visible to users?	<input checked="" type="radio"/> Yes <input type="radio"/> No
Notes	<input type="text"/>