TECHNICAL PAPER

Tips for Configuring the SAS® Viya® Identities Service for LDAP

Last update: January 2022

Contents

Relevant Products and Releases

- SAS® Viya®
    - SAS® Viya® 3.*x* and later releases

Introduction

This paper provides insight into the configuration properties of the SAS Viya Identities service. It describes different scenarios that you might encounter with your LDAP directory server and it offers examples to guide you to a successful configuration. The examples herein are based on common scenarios encountered by customers who have called SAS Technical Support.

The Identities service in SAS Viya retrieves information about identities (of users and groups) from your identity provider. The service also enables the creation and management of custom groups.

The Identities service supports LDAPv3 compliant servers. SAS does not maintain an exhaustive list of supported directory servers, nor does SAS test all directory servers. Therefore, there might be certain features of your LDAP server that are not supported or are not handled as efficiently as with other LDAP servers.

The service works by searching information about groups, users, members, and memberships. It takes the following steps:

- Binds to the LDAP server using the connection properties that are configured for the Identities service. The Identities service supports anonymous or simple bind only. Simple Authentication and Security Layer (SASL) and Kerberos connections to LDAP are not supported.
- Submits an LDAP search request for groups using the search base (**`baseDN`**) **`accountId`**, **`objectClass`**, and **`objectFilter`** values.
- Submits an LDAP search request for users using the search base (**`baseDN`**) **`accountId`**, **`objectClass`**, and **`objectFilter`** values.
- Submits an LDAP search request for user or group information when a client application or service needs member, membership, or other information for an identity. For example, when you select a user on the **Users** page in SAS® Environment Manager, a query is made to LDAP to retrieve the user's memberships. The membership information is displayed on the **Member Of** tab for that user.

The examples in this paper use "ACME" for the company name. The LDAP server address is **`ldap.acme.com`**.

This paper augments the following SAS documentation:

- "Configure Security" in the Linux Deployment Guide of *SAS® Viya® 3.5 Administration*
- "Configure Security" in the Windows Deployment Guide of *SAS® Viya® 3.5 Administration*
- *SAS® Viya® 3.5 Administration: Identity Management*

Minimum Required Configuration

At a minimum, you must set the properties for three configuration instances in order to provide the connection, group, and user information for LDAP. If your LDAP provider is Active Directory, then the minimum settings that you need to change are the properties shown in bold in this section, because the default property values are based on Microsoft Active Directory. For other LDAP providers, such as OpenLDAP, you must provide values for all properties listed here.

- **sas.identities.providers.ldap.connection/**
  - **host**
  - **password**
  - port
  - url
  - **userDN**
- **sas.identities.providers.ldap.group/**
  - accountId
  - **baseDN**
  - distinguishedName
  - member
  - memberOf
  - objectClass
  - objectFilter
  - searchFilter
- **sas.identities.providers.ldap.user/**
  - accountId
  - **baseDN**
  - distinguishedName
  - memberOf
  - objectClass
  - objectFilter
  - searchFilter

Know Your LDAP Server and Data

Before you configure the Identities service, you should gather the following information from your LDAP administrator. These details help you set property values in the Identities service configuration.

- **The name and version of your LDAP provider:** This information becomes important if you experience problems configuring specific attributes. SAS Viya supports LDAPv3 compliant servers. SAS Technical Support has worked with sites that successfully use Microsoft Active Directory, OpenLDAP, Oracle Internet Directory, Azure Active Directory, and others. However, as mentioned earlier, SAS does not maintain an exhaustive list of supported directory servers, nor does SAS test all directory servers.
- **Connection information, including the LDAP host name and port, whether a secure connection is required either through LDAP over SSL (LDAPS) or startTLS, and the search bases for searching for group and user objects:** If you are connecting to Microsoft Active Directory, also find out if you need to connect to the global catalog.

- **The distinguished name of the account that will be used to perform LDAP searches (the bind user) and that account's password, or if you will be using an anonymous connection:** This account must have permission from your LDAP provider to perform searches and to read operational attributes, such as **`memberOf`** and **`isMemberOf`**, that are used for determining group membership.
- **An approximation of the number of users and groups that you expect to "load":** The users and groups are the "identities."
- **All LDAP attributes for a user who will use SAS Viya applications:** Also, if you are loading LDAP groups, you should obtain all attributes for one group.

Confirm That You Can Connect to LDAP and Also Fetch User and Group Entries

After you have all the necessary LDAP connection information, confirm that you can connect and perform searches from the machine that is running the SAS Viya Identities microservice. The `ldapsearch` and `adquery` commands are common ways to connect. You can also use graphical applications such as Softerra LDAP Browser and JXplorer.

In this paper, `ldapsearch` commands are used.

- Run an `ldapsearch` command to return all LDAP attributes and their values for an LDAP group that you want included in SAS Viya. Run a separate command to return memberships to other groups:

```
ldapsearch -H 'ldap://ldap.acme.com:389' -D 'CN=LDAP Query
User,OU=users,DC=acme,DC=com' -w 'Password123' -b
'OU=groups,DC=acme,DC=com' -x -s sub -a always 'CN=Viya Test Group'

ldapsearch -H 'ldap://ldap.acme.com:389' -D 'CN=LDAP Query
User,OU=users,DC=acme,DC=com' -w 'Password123' -b
'OU=groups,DC=acme,DC=com' -x -s sub -a always 'CN=Viya Test Group'
memberOf ismemberOf
```

- Run an `ldapsearch` command to return all LDAP attributes and their values for a user who is a member in the above group. Run a separate command to return memberships to groups. If you are not using LDAP groups, you can instead run the command to return a user that you want included in SAS Viya, regardless of group memberships:

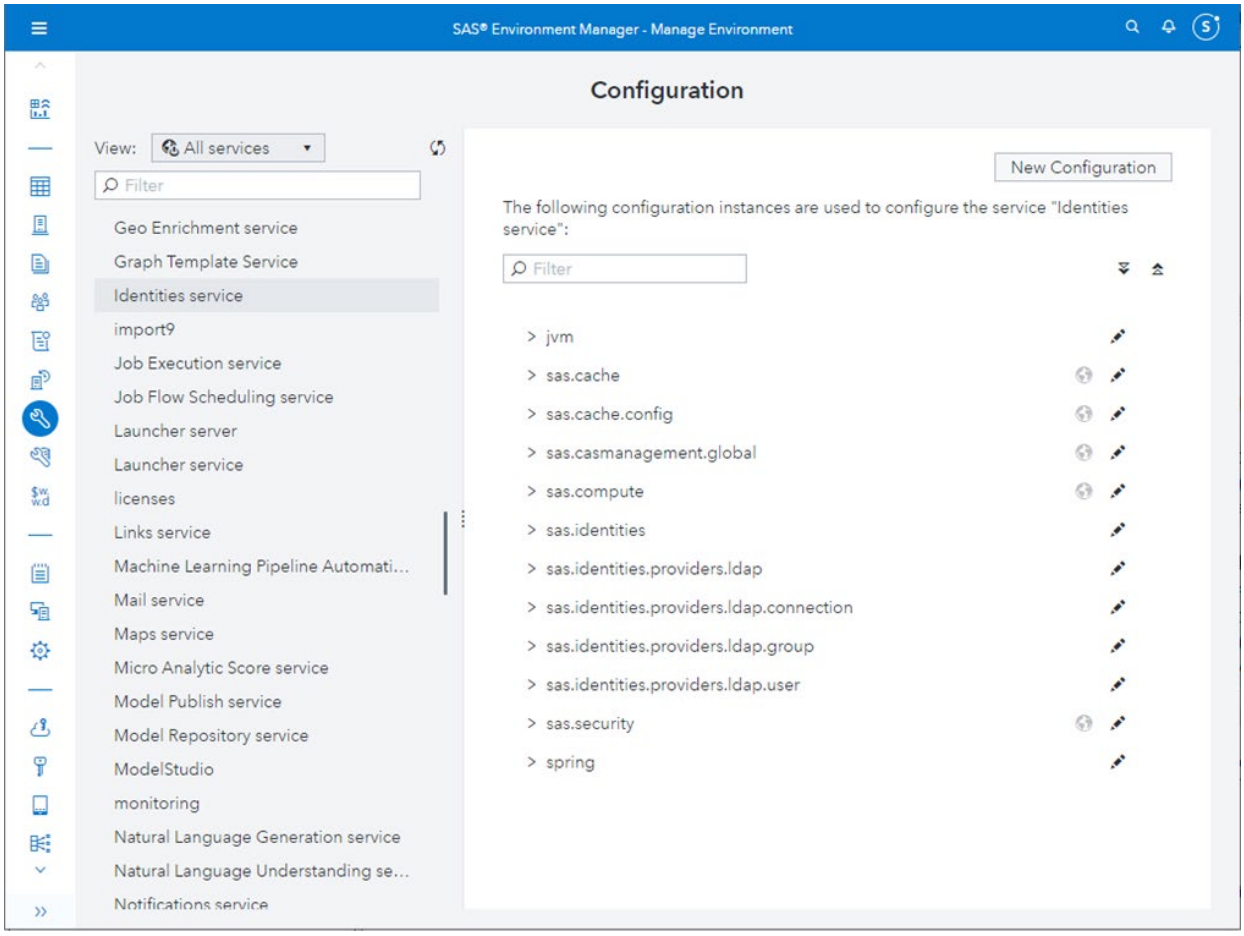
```
ldapsearch -H 'ldap://ldap.acme.com:389' -D 'CN=LDAP Query
User,OU=users,DC=acme,DC=com' -w 'Password123' -b 'OU=users,DC=acme,DC=com'
-x -s sub -a always 'uid=viyauser1'

ldapsearch -H 'ldap://ldap.acme.com:389' -D 'CN=LDAP Query
User,OU=users,DC=acme,DC=com' -w 'Password123' -b 'OU=users,DC=acme,DC=com'
-x -s sub -a always 'uid=viyauser1' memberOf isMemberOf
```

Note: The **`memberOf`** and **`isMemberOf`** attributes are *operational* attributes and are not returned by default by some directory servers. You must therefore request these attributes explicitly in the search query. Your directory server might use a different attribute name, but the attributes **`memberOf`** and **`isMemberOf`** are most common.

Make the Configuration Changes in SAS® Environment Manager

Sign in to SAS Environment Manager using the sasboot account, or an account that is a member of the SAS Administrators custom group. Select the **Configuration** page from the sidebar. (The **Configuration** page is opened by clicking the wrench icon.) Select **All services** from the **View** drop-down menu at the top of the **Configuration** page. Then select **Identities service** from the list. You then see a list of all configuration instances for the Identities service, as shown in the image below. To make a change to any configuration instance, select **Edit**, which is the pencil icon. The configuration then opens in Edit mode. Make the changes and select **Save** to save the changes and close the Edit window. The following sections describe typical configurations for the connection, user, and group configuration instances.



Typical Configurations for an LDAP Connection

The following configuration examples demonstrate how to set the connection properties in different scenarios. The connection properties are made in the **`sas.identities.providers.ldap.connection`** configuration instance.

Here are some tips regarding these connections:

- The port and URL that are shown in the first example are for an unsecured connection from the SAS Viya Identities service to the LDAP server. SAS recommends a secure connection. Ports 636 and 3269 (from the Microsoft Active Directory global catalog) are the default secure ports. When using a secure port, you need to also use the "`ldaps`" protocol for the **`url`** property, as shown in the second example. Another option for a secure connection is to use startTLS, shown in the third example. When you are using startTLS, the port and URL protocol are the same as with an unsecure connection. Set the **`startTLS.mode`** property to **`simple`**.
- You can connect to only one LDAP server. If you have a second server that hosts another domain or that is used for failover, then you should configure a proxy server or load balancer. SAS does not provide instructions for configuring these components, so you should consult with your system administrator.

  In a multi-tenant deployment, you can connect to one LDAP server *per tenant*. To create separate configuration instances for each tenant, turn "on" the property **`Apply to this tenant only (provider)`** in the provider tenant configuration instances. Log in to the tenant with the sasprovider account for that tenant and create new configuration instances.

For an unsecure connection, your settings might look similar to the following:

```
host: ldap.acme.com
password: Password123
port: 389
startTLS.mode: none
url:
ldap://${sas.identities.providers.ldap.connection.host}:${sas.identities.pr
oviders.ldap.connection.port}
userDN: CN=LDAP Query User,OU=users,DC=acme,DC=com
```

This next example shows the properties for a connection using the LDAPS protocol:

```
host: ldap.acme.com
password: Password123
port: 636
startTLS.mode: none
url:
ldaps://${sas.identities.providers.ldap.connection.host}:${sas.identities.p
roviders.ldap.connection.port}
userDN: CN=LDAP Query User,OU=users,DC=acme,DC=com
```

If you are using a secure connection through startTLS, these properties are relevant:

```
host: ldap.acme.com
password: Password123
port: 389
startTLS.mode: simple
url:
ldap://${sas.identities.providers.ldap.connection.host}:${sas.identities.pr
oviders.ldap.connection.port}
userDN: CN=LDAP Query User,OU=users,DC=acme,DC=com
```

Typical Configurations for Group and User Properties

The Identities service makes four types of queries to LDAP:

- for groups
- for users
- for determining a group's members (both users and groups can be members in a group)
- for determining a user's or group's memberships in groups

Each query is generated separately by the Identities service. To fetch users, the Identities service uses only the configuration properties for users. The configuration properties for groups are irrelevant for a user query. Likewise, to fetch groups, the Identities service uses only the configuration properties for groups.

Determining the properties to fetch groups and users is straightforward when you know your data. Properties for group members and memberships are often more difficult to determine, especially for LDAP servers without **`memberOf`** implemented. That is addressed in the examples that follow.

Here are some important tips about the **`accountID`** property value and identity identification:

- The Identities service is a *case-sensitive* service. The user or group is uniquely identified in SAS Viya applications by the case-sensitive value of the **`accountId`** attribute. Therefore, if a user has the **`uid=ViyaUser1`** and that value is later changed in LDAP to **`uid=viyauser1`**, these values are recognized as two different identities. Items created by the identity **`ViyaUser1`** might not be accessible by the identity **`viyauser1`**. This problem results in "lost content" or "lost access" for the end user. Examples are outlined in SAS Note 64250.

  Similarly, for groups, if a credential is saved for the group, or if authorizations are created for the group identity, those rules do not apply to the group if the group **`accountId`** attribute has changed case. For example, **`CN="Marketing"`** is recognized as a different identity from **`CN="marketing"`**.

  Note: Not all services are case-sensitive. This difference might result in unexpected behavior and errors across various services. Refer to SAS Note 68054 for an example.
- The **`accountId`** property value in the user configuration should be set to the LDAP attribute that is used to log in to SAS Viya. Typical LDAP attributes are **`sAMAccountName`**, **`uid`**, **`mail`**, and **`userPrincipalName`**.
- The **`accountId`** property value should not change after users have started accessing the system. For example, if you first set **`accountId`** in the user configuration to **`sAMAccountName`** but then later decide to use **`mail`**, those users who have already logged in to SAS Viya can lose access to their content.
- Likewise, the values of the attribute should not change in LDAP after the user identity has logged in to SAS Viya applications. That results in a new identity, causing the user to lose access to content.
- The values of the **`accountId`** and **`mail`** attributes must be unique across all identities. For example, you cannot have two users with the same **`uid`** property, even if that does not violate any uniqueness constraints in your LDAP. Also, if you are fetching an email attribute from LDAP, the value should be unique for all users. The mail value is used by SASLogon as an external match key.

Configurations for Microsoft Active Directory

The Identities service configuration has default values that are appropriate for Microsoft Active Directory. You generally need to edit only the **baseDN** property for each configuration instance.

The **objectFilter** property value is used to create the LDAP filter that is used to fetch objects. The default property value for groups is very broad, specifying only **(objectClass=group)**. The default **objectFilter** value for users is also very broad, specifying **(&(objectCategory=person)(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))**, which evaluates to all users whose account is not disabled. Therefore, it fetches all objects starting at the search base (**baseDN**) with a scope of **WholeSubtree** (everything below the search base) that have attributes matching the **objectFilter** value. This list can potentially be a very large number of objects and include more groups and users than you really want or need as identities.

The best practice is to limit the number of groups and users in the search query, thereby limiting the number of identities. Here are some ways to do that:

- Set the **baseDN** value to the most restrictive distinguished name (DN) in the directory information tree (DIT) that still contains all groups or users that you want as identities.
- Set the **objectFilter** value to constrain the search even further. The Identities service supports standard LDAP search-filter syntax. A common approach is to use the **memberOf** attribute to get only users who are members of specific groups.

The following tables provide configuration examples for group and user properties for Microsoft Active Directory.

Group Entry in Microsoft Active Directory*	Identities Service Property Values
dn: CN=Viya Test Group,OU=groups,DC=acme,DC=com objectClass: top objectClass: group cn: Viya Test Group member: CN=Viya Demo User 1,OU=users,DC=acme,DC=com member: CN=Viya Demo User 2,OU=users,DC=acme,DC=com member: CN=Viya Demo User 3,OU=users,DC=acme,DC=com memberOf: CN=All Test Groups,OU=groups,DC=acme,DC=com distinguishedName: CN=Viya Test Group,OU=groups,DC=acme,DC=com displayName: Viya Test Group sAMAccountName: svtg	sas.identities.providers.ldap.group/ accountId: sAMAccountName baseDN: OU=groups,DC=acme,DC=com distinguishedName: distinguishedName member: member memberOf: memberOf objectClass: group objectFilter: (&(objectClass=group)(|(CN=Viya Test Group) (CN=Analytics Group))) searchFilter: ${sas.identities.providers.ldap.group.accountId}={0}

* The LDAP entry data that is shown is a subset of attributes that are available.

User Entry in Microsoft Active Directory*	Identities Service Property Values
dn: CN=Viya Demo User1,OU=users,DC=acme,DC=com objectClass: top objectClass: person objectClass: organizationalPerson objectClass: user sAMAccountName: viyauser1 displayName: Viya Demo User1 distinguishedName: CN=Viya Demo User1,OU=users,DC=acme,DC=com memberOf: CN=Viya Test Group,OU=groups,DC=acme,DC=com memberOf: CN=Marketing,OU=groups,DC=acme,DC=com	sas.identities.providers.ldap.user/ accountId: sAMAccountName baseDN: OU=users,DC=acme,DC=com distinguishedName: distinguishedName memberOf: memberOf objectClass: organizationalPerson objectFilter: (&(objectCategory=person)(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2))(|(memberOf=CN=Viya Test Group,OU=groups,DC=acme,DC=com)(memberOf=CN=Analytics Group,OU=groups,DC=acme,DC=com))) searchFilter: ${sas.identities.providers.ldap.user.accountId}={0}

* The LDAP entry data that is shown is a subset of attributes that are available.

Configurations for OpenLDAP and Other LDAP Directory Services

The Identities service configuration has default values for Microsoft Active Directory. However, for other LDAP directory servers, you need to edit the property values to reference the LDAP attributes that your LDAP directory server uses. The attributes depend on your LDAP vendor and how you have implemented it. You can refer to the results from the `ldapsearch` commands that you ran earlier to help choose the appropriate LDAP attributes for each of the Identities service properties.

Be certain to check that you have provided valid LDAP attributes for all the required properties listed in the "Minimum Configuration Required" section above. Any attribute that is not found in LDAP results in a blank value for the corresponding group or user identity. For example, if a user entry in LDAP does not have a **`phone`** attribute, the phone information for that user identity is blank.

Here are some important tips regarding LDAP attributes:

- **`accountId`**: This attribute is the "**`uid`**" for users, and it is the typical LDAP attribute. This attribute value must contain the user name with which the user logs in. The default value of **`sAMAccountName`** is not typical for LDAP servers other than Microsoft Active Directory.
- **`distinguishedName`**: Many LDAP directory servers' DITs do not have an LDAP attribute that is explicitly named **`distinguishedName`**. If that is the case for your LDAP server, then set the **`distinguishedName`** property to **`none`**. Do not set the property to another attribute that contains a distinguished-name value, such as **`dn`**. A common error is to assume that **`dn`** from the `ldapsearch` result is an attribute name. It is a directive in the LDAP Data Interchange Format (LDIF) formatted output, not an

attribute name.

- **memberOf**: This attribute is used to show membership in groups and is relevant for both groups and users. Typical LDAP attributes are **memberOf** and **isMemberOf**. However, the presence of a membership attribute and its name are specific to your LDAP implementation. The value is the distinguished name of the group to which the object is a member.

  Membership attributes are often operational attributes and are not returned by default by some directory servers when you query for all attributes. That can make it difficult to determine the appropriate LDAP attribute to use. First, try to request these two attributes explicitly in the search query, as shown in the `ldapsearch` examples in the "Confirm That You Can Connect to LDAP and Also Fetch User and Group Entries" section above. If neither attribute is returned, your directory server might use a different attribute name, or it is not implemented with a **memberOf** overlay. If your LDAP is not implemented with a **memberOf** overlay, then set the **memberOf** property value to **none**.

  Note: Trace-level logging on **com.sas.identities** shows this behavior in the following ways. This first example shows the logging that occurs when **memberOf** is implemented:

  ```
  Performing operation: query members for group Viya Test Group
  Executing fetch: baseDN='ou=groups,dc=acme,dc=com', filter='(&(cn=Viya Test
  Group)(objectClass=groupOfUniqueNames))'
  Executing basic search: baseDN='ou=groups,dc=acme,dc=com',
  filter='(&(memberOf=cn=Viya Test
  Group,ou=groups,dc=acme,dc=com)(objectClass=groupOfUniqueNames))'
  MemberOf fetch for group members of Group Viya Test Group returned 0 groups
  Executing basic search: baseDN='ou=users,dc=acme,dc=com',
  filter='(&(memberOf=cn=Viya Test
  Group,ou=groups,dc=acme,dc=com)(objectClass=inetOrgPerson)'
  MemberOf fetch for user members of Group Viya Test Group returned 28 users
  ```

  This log example shows what is displayed when **memberOf** is not implemented:

  ```
  Performing operation: query members for group Viya Test Group
  Query members for group Viya Test Group using individual member lookup
  Executing fetch: baseDN='ou=groups,dc=acme,dc=com', filter='(&(cn=Viya Test
  Group)(objectClass=groupOfUniqueNames))'
  Fetching membership via dn lookup for identity
  'uid=viyauser1,ou=users,dc=acme,dc=com
  Fetching membership via dn lookup for identity
  'uid=viyauser2,ou=users,dc=acme,dc=com
  Fetching membership via dn lookup for identity
  'uid=viyauser3,ou=users,dc=acme,dc=com
  [...]
  ```

  When you use an attribute that does not exist, then the **memberOf** column in the User details is (0).

  If your directory server does not have an equivalent attribute to **memberOf**, refer to alternate scenarios in the sections following this one to see your configuration options.

- **objectFilter:** This attribute is used to create the LDAP filter that is used to fetch objects. The default property value for groups is very broad, specifying only **(objectClass=group)**. Modify the **objectFilter** value to use the **objectClass** attribute for groups from your LDAP directory server. Likewise, the default **objectFilter** value for users is also very broad, specifying **(&(objectCategory=person)(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))**. Modify the filter to use the attributes that are appropriate for your LDAP environment.

It is a best practice to limit the number of groups and users in the search query, thereby limiting the number of identities. Here are some ways to do that.

- Set the **baseDN** to the most restrictive DN in the DIT that still contains all groups or users that you want as identities.
- Set the **objectFilter** to constrain the search even further. The Identities service supports standard LDAP search-filter syntax. If your LDAP supports an attribute like **memberOf** or **isMemberOf**, then you can use that attribute to get only users who are members of specific groups.

The following tables provide configuration examples for group and user properties for OpenLDAP.

Group Entry in OpenLDAP	Identities Service Property Values
dn: cn=Viya Test Group,ou=groups,dc=acme,dc=com objectClass: groupOfUniqueNames cn: Viya Test Group displayName: SAS Viya Test Group memberOf: cn=All Test Groups,ou=groups,dc=acme,dc=com uniqueMember: uid=viyauser1,ou=users,dc=acme,dc=com uniqueMember: uid=viyauser2,ou=users,dc=acme,dc=com uniqueMember: uid=viyauser3,ou=users,dc=acme,dc=com	sas.identities.providers.ldap.group/ accountId: cn baseDN: ou=groups,dc=acme,dc=com distinguishedName: none member: uniqueMember memberOf: memberOf objectClass: groupOfUniqueNames objectFilter: (objectClass=groupOfUniqueNames) searchFilter: ${sas.identities.providers.ldap.group.accountId}={0}

User Entry in OpenLDAP	Identities Service Property Values
dn: uid=viyauser1,ou=users,dc=acme ,dc=com objectClass: person objectClass: organizationalPerson objectClass: user uid: viyauser1 displayName: Viya Demo User1 memberOf: cn=SAS Viya Test Group,ou=groups,dc=acme,d c=com memberOf: cn=Marketing,ou=groups,dc=acme ,dc=com	sas.identities.providers.ldap.user/ accountId: uid baseDN: ou=users,dc=acme,dc=com distinguishedName: none memberOf: memberOf objectClass: organizationalPerson objectFilter: (objectClass=organizationalPerson) searchFilter: ${sas.identities.providers.ldap.user.accoun tId}={0}

* The LDAP entry data that is shown is a subset of attributes that are available.

Determining Members and Memberships without a memberOf Attribute

If your LDAP implementation does not have the equivalent of the **memberOf** attribute, then the Identities service uses a recursive lookup approach to determine members and memberships. The following examples are alternate configurations that you can use to compensate for the lack of a **memberOf** attribute.

Scenario 1

This scenario is defined by the following conditions:

- A user's group membership is based *only* on an attribute on the group object. From the perspective of the user object, there is no attribute that you can query to get the groups to which the user is a member.
- The member attribute on the group object is a relative distinguished name (RDN) value, such as **viyauser1**, and not a DN value such as **uid=viyauser1,ou=users,dc=acme,dc=com**.
- The member attribute value on the group object must match the value of the **accountID** attribute on the user object. For example, if the **accountID** property for a user is **uid**, then the member attribute value on the group object must match the **uid** attribute value on the user object, as shown in the following table.

Here is a note on configuration settings:

- Set the **memberOf** property to **none**. Refer to SAS Note 61750.

Group Entry in OpenLDAP	Identities Service Property Values
dn: cn=Viya Test Group,ou=groups,dc=acme,dc=com objectClass: posixGroup cn: Viya Test Group gidNumber: 505 memberUid: viyauser1 memberUid: viyauser2 memberUid: viyauser3	sas.identities.providers.ldap.group/ accountId: cn baseDN: ou=groups,dc=acme,dc=com distinguishedName: none member: memberUID memberOf: none objectClass: posixGroup objectFilter: (objectClass= posixGroup) searchFilter: ${sas.identities.providers.ldap.group.accountId}={0}

User Entry in OpenLDAP	Identities Service Property Values
dn: uid=viyauser1,ou=users,dc=acme,dc=com objectClass: person objectClass: posixAccount cn: Viya Demo User1 sn: Viya Demo User1 gidNumber: 8000 uid: viyauser1 uidNumber: 501	sas.identities.providers.ldap.user/ accountId: uid baseDN: ou=users,dc=acme,dc=com distinguishedName: none memberOf: none objectClass: person objectFilter: (objectClass=person) searchFilter: ${sas.identities.providers.ldap.user.accountId}={0}

* The LDAP entry data that is shown is a subset of attributes that are available.

Note: Support for an RDN value for **member** and **memberOf** properties began with Identities service version 2.13.21 in SAS® Viya® 3.4. Refer to SAS Note 61750.

Scenario 2

This scenario is defined by the following conditions:

- A user's group membership is based *only* on an attribute on the group object. From the perspective of the user

object, there is no attribute that you can query to get the groups to which the user is a member.

- The member attribute on the group object is a DN value.

Here is a note on configuration settings:

- This scenario is like Scenario 1, but in this case the **member** attribute is a DN value, and the **cn** is an RDN value. Since the member attribute value on the group object **must** match the value of the **accountID** attribute on the user object, this scenario is not supported.

Group Entry and Attribute Name and Value	Identities Service Property Name and Value
dn: cn=Viya Test Group,ou=groups,dc=acme,dc=com objectclass: groupOfNames cn: Viya Test Group member: cn=viyauser1,ou=users,dc=acme,dc=com member: cn=viyauser2,ou=users,dc=acme,dc=com member: cn=viyauser3,ou=users,dc=acme,dc=com	sas.identities.providers.ldap.group/ accountId: cn baseDN: ou=groups,dc=acme,dc=com distinguishedName: none member: member memberOf: none objectClass: groupOfNames objectFilter: (objectClass=groupOfNames) searchFilter: ${sas.identities.providers.ldap.group.accountId}={0}

User Entry and Attribute Name and Value	Identities Service Property Name and Value
dn: cn=viyauser1,ou=users,dc=acme,dc=com objectClass: person cn: viyauser1 sn: Viya Demo User 1	sas.identities.providers.ldap.user/ accountId: cn baseDN: ou=users,dc=acme,dc=com distinguishedName: none memberOf: none objectClass: person objectFilter: (objectClass=person) searchFilter: ${sas.identities.providers.ldap.user.accountId}={0}

Scenario 3

This scenario is defined by the following conditions:

- The group membership for a user is based on the **gidNumber** attribute and another **member** or **memberOf**

attribute.

- The **member** or **memberOf** attribute for the user or group object must follow one of the supported scenarios above.

Here are some notes on configuration settings:

- Set the **primaryGroupMembershipsEnabled** property to **true**.
- Set the **member** and **memberOf** properties based on one of the supported scenarios above.

The following tables provide configuration examples for group and user properties for OpenLDAP.

Group Entry in OpenLDAP	Identities Service Property Values
dn: cn=Viya Test Group,ou=groups,dc=acme,dc=com objectClass: posixGroup cn: Viya Test Group gidNumber: 9000 memberUid: viyauser1 memberUid: viyauser2	sas.identities.providers.ldap.group/ accountId: cn baseDN: ou=groups,dc=acme,dc=com distinguishedName: none member: memberUID memberOf: none objectClass: posixGroup objectFilter: (objectClass=posixGroup) searchFilter: ${sas.identities.providers.ldap.group.accountId}={0}

User Entry in OpenLDAP	Identities Service Property Values
dn: uid=viyauser1,ou=users,dc=acme,dc=com objectClass: person objectClass: posixAccount cn: Viya Demo User1 sn: Viya Demo User1 gidNumber: 9000 uid: viyauser1 uidNumber: 5001 dn: uid=viyauser3,ou=users,dc=acme,dc=com objectClass: person	sas.identities.providers.ldap.user/ accountId: uid baseDN: ou=users,dc=acme,dc=com distinguishedName: none memberOf: none objectClass: person objectFilter: (objectClass=person) searchFilter: ${sas.identities.providers.ldap.user.accountId}={0} sas.identities.providers.ldap/ primaryGroupMembershipsEnabled: on

objectClass: posixAccount cn: Viya Demo User3 sn: Viya Demo User3 gidNumber: 9000 uid: viyauser3 uidNumber: 5003	

* The LDAP entry data that is shown is a subset of attributes that are available.

Note: This scenario is documented in SAS Note 62464.

FAQ Section

How Is the Identities Service Cache Used?

The cache contains only summary information (such as IDs, names, descriptions, and so on) for users and groups. It contains custom groups but does not contain LDAP group membership information. The sole purpose of the cache is to provide an alternative mechanism for searching for a collection of users or groups in order to improve performance: for example, when SAS Environment Manager displays the list of users. Anytime a client needs to fetch more information about a user (for example, their membership information), the Identities service requests the data directly from LDAP. It is recommended that you leave the cache enabled. Set the property in the **sas.identities** configuration instance.

When Should I Disable pagedResults?

This property instructs the Identities service to add the **pagedResults** control to its search requests. That control enables you to set the number of entries returned per page and is a benefit if the search request could potentially return a large number of entries (say, over 1,000). The **pagedResults** property is set to **on** by default. An error like the following in the Identities service log indicates that your LDAP server does not support the **pagedResults** control.

```
DEBUG 15526 --- [    bootstrap-1] o.s.l.p.v.DefaultDirContextValidator
: service [a23470ea0ac829b6] DirContext
'javax.naming.ldap.InitialLdapContext@2551795' failed validation with
an exception.[LDAP: error code 12 - The search request cannot be
processed because it contains a critical control with OID
1.2.840.113556.1.4.319 that is not supported by the Directory Server
for this type of operation].
```

If you get this error, then change the **pagedResults** property in the **sas.identities.providers.ldap** configuration instance to **off**.

When Should I Enable followReferrals?

This property instructs the Identities service to search continuation references that are returned by the LDAP server. The option is often needed when the **baseDN** is set to a domain (such as **DC=acme,DC=com**) rather than an object higher in the LDAP hierarchy (such as **OU=users,DC=acme,DC=com**) and you have more than one domain in your LDAP forest. You might also see a message like the following in the Identities service log:

```
org.springframework.ldap.PartialResultException: Unprocessed Continuation
Reference(s); nested exception is javax.naming.PartialResultException:
Unprocessed Continuation Reference(s); remaining name 'DC=acme,DC=com'
```

If you get this error, then change the **followReferrals** property in the **sas.identities.providers.ldap** configuration instance to **on**.

More information about this topic is in SAS Note 61467.

When Should I Use the Microsoft Active Directory Global Catalog?

In a multi-domain forest environment in Microsoft Active Directory, you can configure the LDAP connection to use the global catalog server. This setting can provide a more efficient search result because the global catalog server stores a replica of a subset of attributes for objects across all domains in the forest. No referrals are needed. If you are receiving the error from the section above, the global catalog is a way to avoid the referral error. The default port for the Active Directory global catalog server is 3268. For LDAPS, the default global catalog port is 3269.

How Can I Load Nested LDAP Groups?

If your LDAP provider supports the object identifier (OID) 1.2.840.113556.1.4.1941 (LDAP_MATCHING_RULE_IN_CHAIN), then you can integrate it into the **objectFilter** for users or groups.

Suppose that Viya Test Group is a subgroup (a member of) the All Test Groups group, as shown in the configuration examples for the group and user properties for Microsoft Active Directory above.

To load the All Test Groups group and all groups that are a member of the All Test Groups group, modify **sas.identities.providers.ldap.group/objectFilter** to include the filter for the All Test Groups group and the **memberOf** attribute with the LDAP_MATCHING_RULE_IN_CHAIN extended match operator on the All Test Groups group:

```
(&(objectclass=group) (|(CN=All Test Groups)
(memberOf:1.2.840.113556.1.4.1941:=CN=All Test
Groups,OU=groups,DC=acme,DC=com)))
```

To load all users who are direct members of the All Test Groups group, and all users who are members of the groups that are nested under the All Test Groups group, such as the Viya Test Group group, modify the **sas.identities.providers.ldap.user/objectFilter** to include the LDAP_MATCHING_RULE_IN_CHAIN extended match operator:

```
(&(objectCategory=person)(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2))(|(memberOf=CN=All Test Groups,OU=groups,DC=acme,DC=com)(memberOf:1.2.840.113556.1.4.1941:=CN=All Test Groups,OU=groups,DC=acme,DC=com)))
```

What Other Settings Can Potentially Improve Performance?

The Identities service maintains a pool of connections to the LDAP server. Pooling is enabled by default, maintaining a maximum of eight connections in the pool. If you experience delays in retrieving identity data, the LDAP pool might be exhausted. If the pool is exhausted, you see an error similar to the following in the Identities service log:

```
WARN 17019 --- [auto-1-exec-282] c.s.i.p.l.LdapIdentityQueryRepository    : sas.logon [b184c35bbadec502] [IDENTITY FETCH LDAP ERROR] Error occurred while fetching identity: Failed to borrow DirContext from pool.; nested exception is java.util.NoSuchElementException: Timeout waiting for idle object
```

You can modify pooling settings in the **`sas.identities.providers.ldap.connection`** configuration instance. The "Tuning: LDAP Connection Pool on Linux" section in *SAS® Viya® 3.5 Administration* contains some very basic tuning information.

Conclusion

With insight into the configuration properties that this paper has provided, and the illustration of different scenarios that you might encounter with your LDAP directory server, you are now ready to configure the SAS Viya Identities service with confidence. If you experience errors, refer to SAS Note 61882 for troubleshooting steps.

Release Information Content Version: 2.0 January 2022.

Trademarks and Patents SAS Institute Inc. SAS Campus Drive, Cary, North Carolina 27513

To contact your local SAS office, please visit: sas.com/offices