Paper SAS4315-2020

SAS® Viya® 3.5 Kerberos Constrained Delegation: Putting the Dog on a Leash

Stuart J Rogers, SAS Institute Inc.

ABSTRACT

Do you maintain a SAS® Viya® 3.5 environment leveraging Kerberos for authentication? Are you struggling with the strict security requirements of your IT department? SAS Viya 3.5 introduces support for Kerberos constrained delegation. This paper outlines the additional steps you must complete to correctly implement Kerberos constrained delegation with your SAS Viya 3.5 environment. This implementation enables you to leverage technologies such as Microsoft Windows Defender Credential Guard on your client desktops. Learn about the configuration changes you need to make and learn some effective troubleshooting techniques to get your environment working and make your users happy.

INTRODUCTION

Security and the protection of user credentials is a major concern for all businesses. While the Kerberos authentication protocol provides a strong authentication mechanism for networked services, there still exists several attack vectors that can be exploited. Microsoft has provided several enhancements to mitigate these attack vectors. First, Microsoft introduced Kerberos constrained delegation with Microsoft Windows Server 2003 (Microsoft 2018). Then, Microsoft provided further options with Windows Server 2012 and resource-based constrained delegation (Microsoft 2018). Finally, with Windows 10 and Windows Server 2016 Microsoft introduced Windows Defender Credential Guard to protect local credentials and enforce constrained delegation (Microsoft 2016).

SAS Viya 3.5 introduces support for Kerberos constrained delegation. Kerberos constrained delegation can be used to authenticate to both SAS Cloud Analytic Services (CAS) and SAS Compute Server from the SAS Viya 3.5 web applications. Equally, Kerberos constrained delegation can be used from CAS to authenticate to data sources or from SAS Compute Server to CAS or data sources. Kerberos constrained delegation can even be used to authenticate through a SAS 9.4 environment, to CAS and onto your data source.

This paper will explain, at a high level, what Kerberos constrained delegation is and how it works with SAS Viya 3.5. We will also look at the prerequisites you need to complete and how you can configure Kerberos constrained delegation with SAS Viya 3.5. Finally, we will examine some troubleshooting techniques you can leverage to ensure Kerberos constrained delegation is operating correctly.

DESCRIPTION OF CONSTRAINED DELEGATION

Constrained delegation differs from unconstrained delegation in the following ways:

- The service does not require the user to forward either the Ticket-Granting Ticket (TGT) or the proxy ticket.
- The user does not need to authenticate using Kerberos and the user does not need to have a TGT or a proxy service ticket.
- Windows Local Group policy can be used to limit the services that can be delegated.

- The client has no control over whether a service can delegate on behalf of the user. The client does not request delegation, nor does it pass a forwardable TGT to the service.
- The client cannot detect that delegation will be, or has been, performed.

HIGH-LEVEL OVERVIEW OF KERBEROS CONSTRAINED DELEGATION

Kerberos constrained delegation or Service for User (S4U) is a Microsoft extension to the Kerberos protocol. S4U provides two extensions to the Kerberos protocol. Together, these extensions allow a service to obtain a Kerberos service ticket on behalf of a user. The first extension is Service for User to Proxy (S4U2proxy) that allows a service to obtain a service ticket on behalf of a user to a different service. The second extension is Service for User to Self (S4U2self) that allows a service to obtain a Kerberos service ticket to itself. The S4U extension is defined in a separate specification document (Microsoft 2018).

There are two different types of Kerberos constrained delegation, as illustrated in Figure 1. "Traditional" constrained delegation was introduced in Microsoft Windows Server 2003. With this type of constrained delegation, the constraints are defined against the front-end service and govern what back-end services the front-end service can delegate credentials to. Both the front-end and back-end services must be in the same Kerberos realm. "Traditional" constrained delegation is now supported on multiple operating systems and the constraints can be configured either in GUI tools or with Windows PowerShell command-lets.

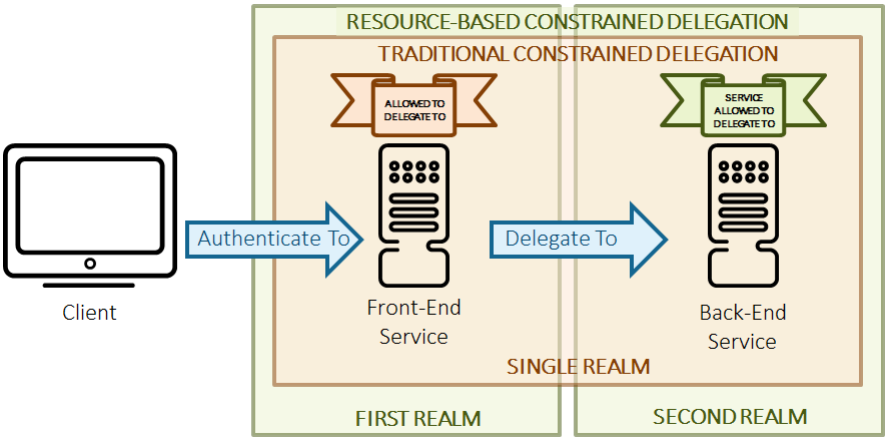

Figure 1. "Traditional" Versus Resource-Based Constrained Delegation

The second type of Kerberos constrained delegation is resource-based constrained delegation and was introduced in Microsoft Windows Server 2012. With resource-based constrained delegation, the constraint is configured on the back-end service. This enables the resource owner of the back-end service to configure what front-end services can delegate credentials to the back-end service. The front-end services and back-end service do not have to be in the same Kerberos Realm with resource-based constrained delegation. Resource-based constrained delegation requires at least one domain controller running Windows Server 2012 and higher. Additionally, it requires the front-end service to also be running on Windows Server 2012 and higher, while the back-end service could be running on Windows Server 2003 and higher. Support for resource-based constrained delegation is being added to the MIT Kerberos libraries. At the time of writing, that support has not been released. Finally, resource-based constrained delegation can only be configured using Windows PowerShell command-lets.

S4U2SELF

The S4U2self extension allows a service to obtain a service ticket to itself on behalf of a user. This is referred to as protocol transition, since this enables a user to be authenticated using a mechanism other than Kerberos. Then, Kerberos is used for connections from the front-end service to back-end services. Figure 2 details the process flow for the S4U2self extension.

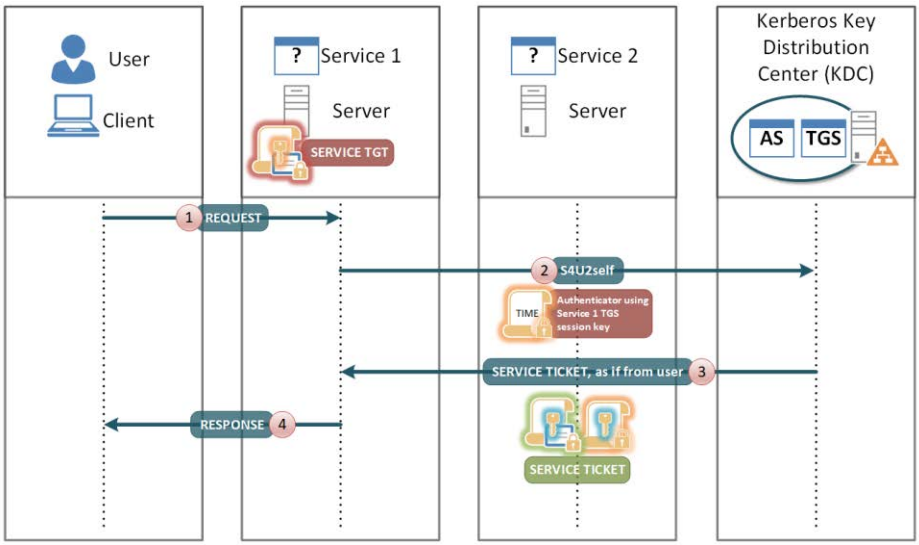

Figure 2. S4U2self Process Flow

The following steps are followed:

1. The user's machine makes a request to Service 1. The user is authenticated, but Service 1 does not have the user's authorization data. Typically, this is due to the authentication being performed by some means other than Kerberos.
2. Service 1, which has already authenticated with the key distribution center (KDC) and has obtained its TGT, asks for a service ticket to itself on behalf of the named user by the S4U2self extension. The user is identified by the username and the user's realm name in the S4U2self data. Alternatively, if Service 1 is in possession of the user's certificate, it can use the certificate to identify the user to the KDC using the PA-S4U-X509-USER structure.
3. The KDC returns a service ticket addressed to Service 1 as if it had been requested from the user with the user's own TGT. The service ticket might contain the authorization data of the user.
4. Service 1 can use the authorization data from the service ticket to fulfill the user's request. The service then responds to the user. Although S4U2self provides information about the user to Service 1, this extension does not allow Service 1 to make requests of other services on the user's behalf. That is the role of S4U2proxy.

Protocol transition or S4U2self relies on the setting `TrustedToAuthForDelegation`; which is a Boolean setting to control whether the KDC sets the FORWARDABLE ticket flag in S4U2self service tickets for principals for the service. S4U implementations that use Active Directory for the account database SHOULD use the `userAccountControl` attribute TA flag.

S4U2PROXY

The S4U2proxy extension provides a service that obtains a service ticket to another service on behalf of a user. This feature is known as constrained delegation. The same process flow is followed for both "traditional" and resource-based constrained delegation. The differences are in the attributes that limit or constrain the delegation. Figure 3 details the process flow for the S4U2proxy extension.

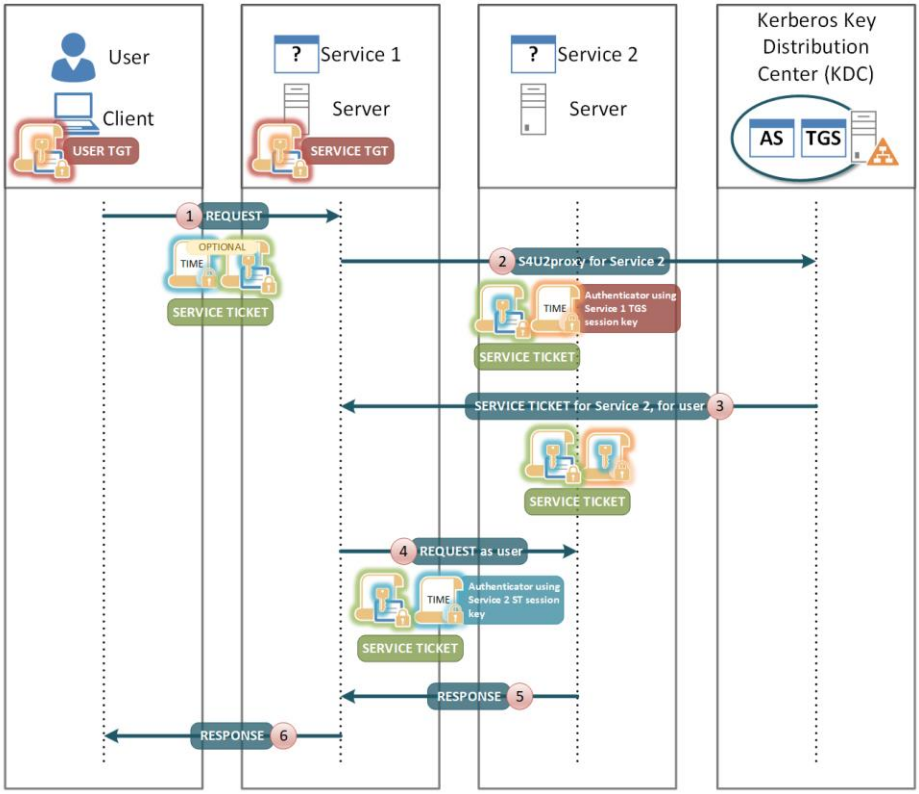

Figure 3. S4U2proxy Process Flow

The following steps are followed:

1. The user's machine makes a request to Service 1. Service 1 needs to access resources on Service 2 as the user. However, Service 1 does not have a forwarded TGT from the user to perform delegation by a forwarded TGT. Two preconditions apply to this step:
   - Service 1 has already authenticated with the KDC and has a valid TGT.
   - Service 1 has a forwardable service ticket from the user to Service 1. This forwardable service ticket might have been obtained by a KRB_AP_REQ message, as specified in [RFC4120] section 3.2 or by an S4U2self request.
2. Service 1 requests a service ticket to Service 2 on behalf of the named user. The user is identified by the client name and the client realm in the service ticket for Service 1. The authorization data in the ticket to be returned is also copied from the service ticket.
3. If a privilege attribute certificate (PAC) is in the request, the KDC validates the PAC by checking the signature data of the PAC structure. If the PAC is valid, or not present, the KDC returns a service ticket for Service 2, but the client identity stored in the cname and crealm fields of the service ticket are that of the user, not Service 1.

4. Service 1 uses the service ticket to make a request to Service 2. Service 2 treats this request as coming from the user and assumes that the user was authenticated by the KDC.
5. Service 2 responds to the request.
6. Service 1 responds to the user's initial request shown in message 1.

"Traditional" constrained delegation relies on the ServicesAllowedToSendForwardedTicketsTo setting, which is a list of services to which a service will be allowed to forward tickets to support constrained delegation. S4U implementations that use Active Directory for the configuration database should use the msDS-AllowedToDelegateTo attribute. Resource-based constrained delegation relies on ServicesAllowedToReceiveForwardedTicketsFrom, which is a SECURITY_DESCRIPTOR that specifies from which services a service will accept forwarded service tickets. S4U implementations that use Active Directory for the configuration database should use the msDS-AllowedToActOnBehalfOfOtherIdentity attribute. All types of delegation respect the DelegationNotAllowed setting, which is a Boolean setting to prevent PROXIABLE or FORWARDABLE ticket flags in tickets for the principal. Implementations that use Active Directory for the account database should use the userAccountControl attribute ND flag.

HOW CONSTRAINED DELEGATION WORKS WITH SAS VIYA 3.5

SAS Viya 3.5 supports Kerberos constrained delegation in the following scenarios:

1. SAS Logon Manager authenticating end-users to the SAS Viya visual applications
2. CAS launching sessions, and accessing data
3. SAS Launcher Service and Compute Server, launching sessions, accessing SAS Cloud Analytic Services, and accessing data

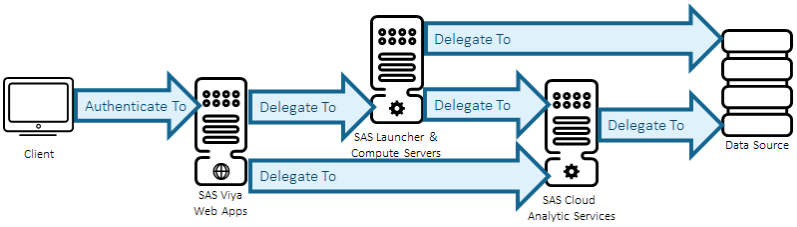

Figure 4. Overview of Constrained Delegation with SAS Viya 3.5

Your end-users accessing SAS Viya visual applications authenticate with SAS Logon Manager. SAS Logon Manager processes Kerberos authentication to authenticate your end-users. The services within SAS Viya web applications perform S4U2self requests to obtain a service ticket for itself on behalf of end-users as part of connecting to CAS or SAS Compute Server. CAS Controller leverages the delegated credentials to launch the CAS session for end-users and the CAS session uses delegated credentials to access data sources. SAS Launcher Service uses the delegated credentials to launch a SAS Compute Server session. The SAS Compute Server session then uses the delegated credentials to access data sources. Figure 4 presents this high-level overview of how Kerberos constrained delegation is used in SAS Viya 3.5.

STANDARD PROCESS FOR KERBEROS CONSTRAINED DELEGATION

Once your end-users are authenticated to SAS Logon Manager using Kerberos, SAS Viya visual applications can launch a CAS session using constrained delegation. Or SAS Launcher Service can launch a SAS Compute Server session using constrained delegation. Within the

SAS Compute Server session, constrained delegation can then be used to access a data source directly or launch a CAS session.

A good example to illustrate the process that occurs using Kerberos constrained delegation is the use-case of your end-user logging into SAS Studio 5.2 (Enterprise), launching a CAS session, and then accessing data in Microsoft SQL Server. So first we will examine launching SAS Studio 5.2 (Enterprise), as shown in Figure 5.

The following steps are followed:

1. The user authenticates using Kerberos to SAS Logon Manager and requests SAS Studio 5.2 (Enterprise).
2. SAS Launcher Service (microservice) triggers a S4U2self request.
3. The S4U2self response is a service ticket for HTTP as if from the user.
4. SAS Launcher Service (microservice) triggers a S4U2proxy request using the service ticket from the S4U2self response.
5. The S4U2proxy response contains the service ticket for sas-launcher for the user.
6. SAS Launcher Service (microservice) uses the service ticket for sas-launcher to start the SAS Compute Server session as the user.
7. SAS Compute Server responds to SAS Compute Service.
8. SAS Studio 5.2 (Enterprise) responds to the user.

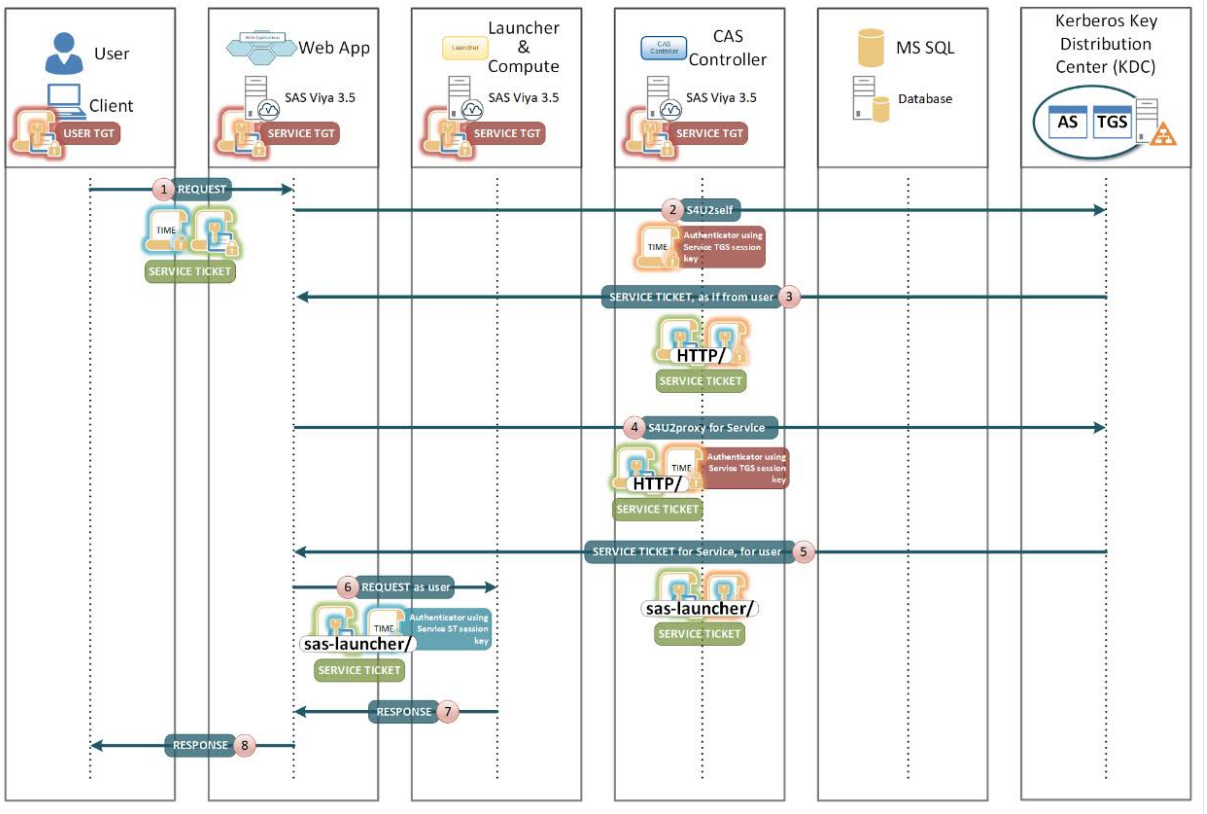

Figure 5. Example: Launching SAS Studio 5.2 (Enterprise)

Then, your end user within SAS Studio 5.2 (Enterprise) enters and submits the SAS code to launch a CAS session, as shown in Figure 6.

The following steps are followed:

1. The user enters code in SAS Studio 5.2 (Enterprise): "cas mysession;".
2. SAS Studio 5.2 (Enterprise) submits code to the SAS Compute Server session requesting the launch of a CAS session.
3. SAS Compute Server session triggers a S4U2proxy request using the service ticket acquired during session launch.
4. The S4U2proxy response contains the service ticket for sascas for the user.
5. SAS Compute Server session uses the service ticket for sascas to start the CAS session as the user.
6. The CAS session responds to the SAS Compute Server session.
7. SAS Compute Server session responds to SAS Studio 5.2 (Enterprise).
8. SAS Studio 5.2 (Enterprise) responds to the user.

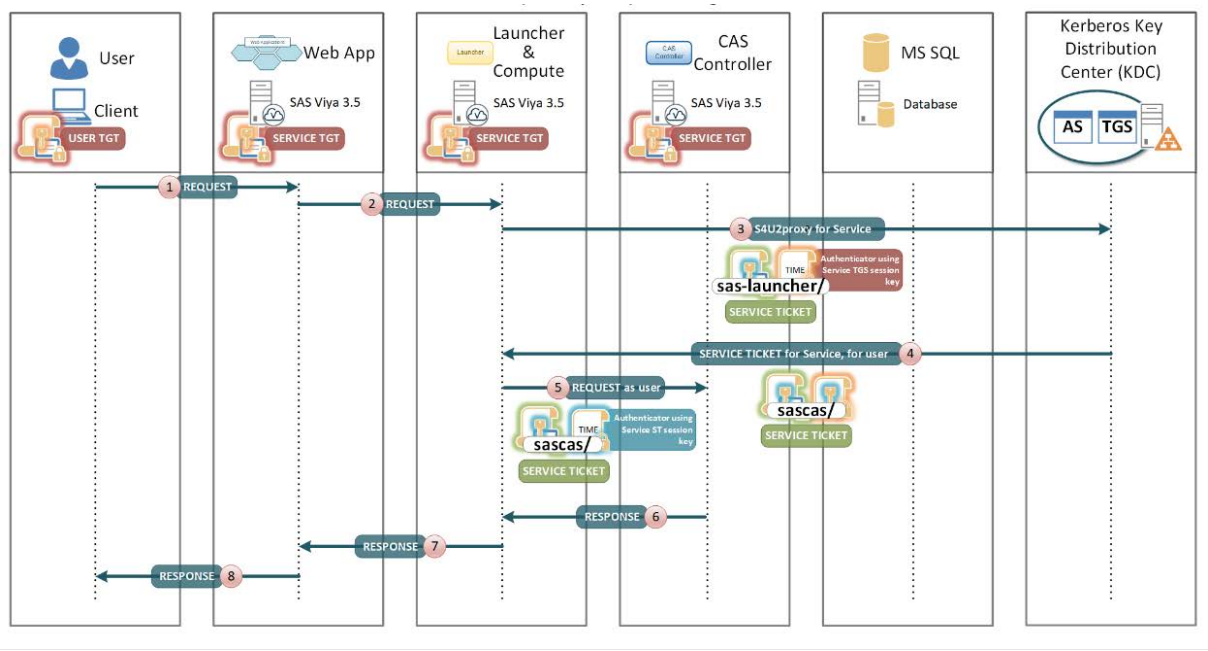

Figure 6. Example: SAS Studio 5.2 Launching CAS Session

Finally, with their CAS session running, the end user enters and submits SAS code in SAS Studio 5.2 (Enterprise) to cause CAS to process data from Microsoft SQL Server, as illustrated in Figure 7.

The following steps are followed:

1. The user enters code requesting CAS to process data.
2. SAS Studio 5.2 (Enterprise) submits code to the SAS Compute Server session.
3. SAS Compute Server session requests CAS session to process data.
4. CAS session controller triggers a S4U2proxy request using the service ticket acquired during session launch.
5. The S4U2proxy response contains the service ticket for MSSQLSvc for the user.

6. CAS session controller uses the service ticket for MSSQLSvc to authenticate and request data from MS SQL Server.
7. MS SQL Server responds to the CAS session.
8. CAS session processes data and responds to the SAS Compute Server session.
9. SAS Compute Server session responds to SAS Studio 5.2 (Enterprise).
10. SAS Studio 5.2 (Enterprise) responds to the user.

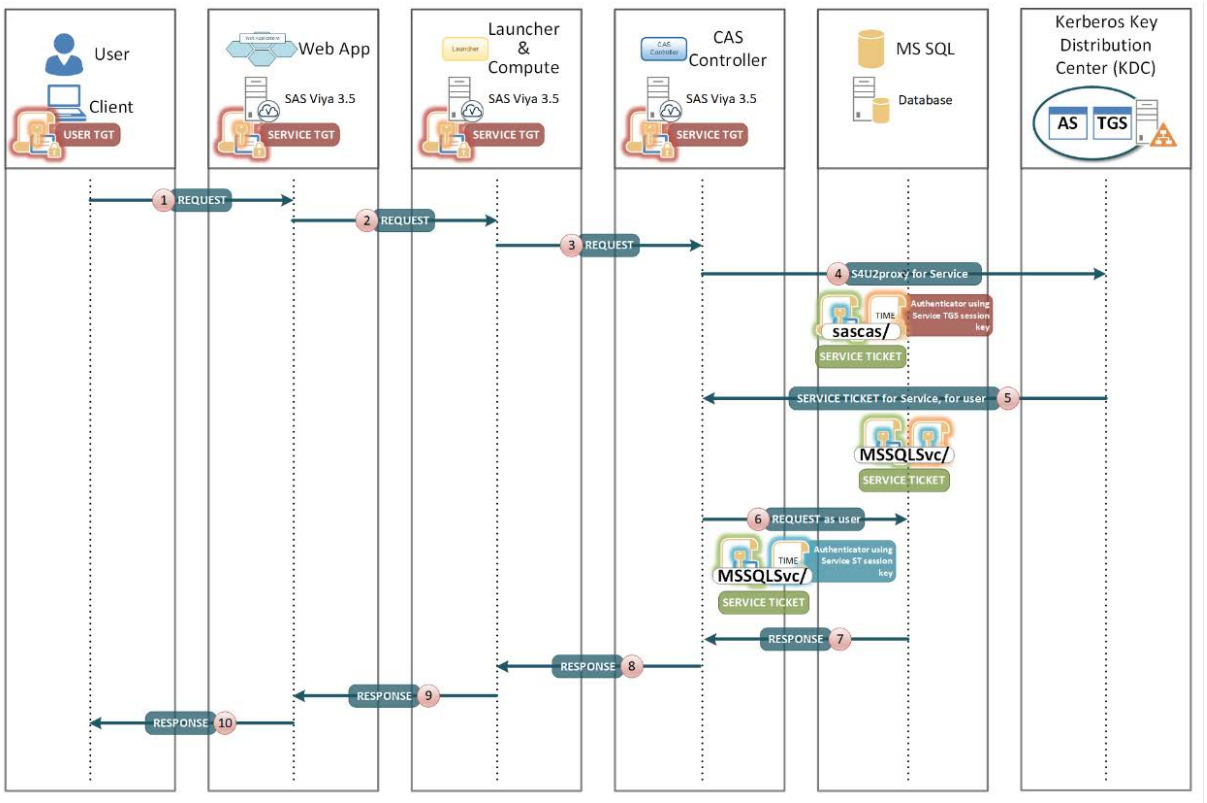

Figure 7. Example: CAS Session Accessing Data in Microsoft SQL Server

This use-case illustrates Kerberos constrained delegation with SAS Viya 3.5, where the microservices leverage S4U2self requests to obtain the Kerberos service ticket for HTTP. Then S4Uproxy requests are used to acquire service tickets for sas-launcher, sascas, and MSSQLSvc.

OTHER USE-CASES FOR KERBEROS CONSTRAINED DELEGATION

In the previous section, we have presented the most common use-case for Kerberos constrained delegation. However, Kerberos constrained delegation can also be used in other scenarios.

You can use Kerberos constrained delegation to combine a SAS 9.4 M6 environment with CAS. You will require additional hot fixes for your SAS 9.4 M6 environment as explained in the following SAS Note: http://support.sas.com/kb/63/143.html. Then you can use Kerberos constrained delegation to authenticate from your SAS sessions in SAS 9.4 directly to CAS.

Kerberos constrained delegation can be used within your SAS Viya 3.5 environment and out to data sources even when your end-users do not use Kerberos to authenticate to SAS Logon Manager, since the microservices can use S4U2self requests to obtain a service ticket for HTTP. This requires that the username in SAS Viya 3.5 matches the Kerberos username for your end-users and that the end-users are in the same Kerberos realm as the HTTP principal. By default, when using Active Directory as your LDAP provider for SAS Viya 3.5, the SAS Viya username will be the sAMAccountName of your end-users. As such, so long as your end-users authenticate to SAS Logon Manager with their sAMAccountName, either with username and password, or SAML, or OpenID Connect, then the S4U2self requests will complete successfully.

SAS Viya 3.5 supports direct connections from SAS Enterprise Guide 8.2 to SAS Object Spawner. This direct connection does not require a SAS Metadata Server and enables your end-users to directly submit SAS code to the SAS Foundation instance that is part of your SAS Viya 3.5 environment. Currently SAS Enterprise Guide 8.2 can leverage Kerberos constrained delegation to authenticate direct connections to SAS Viya Object Spawner only when TLS is disabled on the Object Spawner. This is the default setup for SAS Viya Object Spawner on Microsoft Windows or a programming-only deployment on Linux. However, on a Linux full deployment, the configuration of SAS Viya Object Spawner must be updated to disable TLS. Encryption in-motion is still available on the connection between SAS Enterprise Guide 8.2 and SAS Viya Object Spawner. In this case, it will use SAS/SECURE with the AES encryption algorithm.

SUPPORT FOR DATA SOURCES WITH KERBEROS CONSTRAINED DELEGATION

SAS Viya 3.5 supports Kerberos constrained delegation, however not all data sources support Kerberos constrained delegation. Both Microsoft SQL Server and Teradata support Kerberos constrained delegation. Thus, Kerberos constrained delegation can be used with SAS/ACCESS and data connectors for both Microsoft SQL Server and Teradata. In addition, SAS Data Connect Accelerator for Teradata can leverage Kerberos constrained delegation.

At the time of writing, due to third-party functionality that does not yet support constrained delegation, these SAS products do not currently support Kerberos constrained delegation functionality:

- SAS/ACCESS® Interface to Hadoop
- SAS/ACCESS® Interface to Impala
- SAS/ACCESS® Interface to JDBC
- SAS/ACCESS® Interface to Oracle
- SAS® Plug-ins for Hadoop

SAS VIYA 3.5 SUPPORT FOR RESOURCE-BASED CONSTRAINED DELEGATION

Resource-based constrained delegation is only supported by Microsoft Windows. The MIT Kerberos libraries, at the time of writing, do not support resource-based constrained delegation. This means that resource-based constrained delegation can only be supported on SAS Viya 3.5 running on Microsoft Windows. "Traditional" constrained delegation is supported on both Microsoft Windows and Linux.

PREREQUISITES FOR CONSTRAINED DELEGATION

Now that we have examined how Kerberos constrained delegation operates, we can move on to discuss the prerequisites.

KERBEROS

First and foremost, your SAS Viya 3.5 environment must be configured for Kerberos authentication. We will not repeat all the standard Kerberos prerequisites and configuration settings in this paper. The paper "Extending the Reach of Kerberos Authentication from SAS® Viya® 3.4 to Apache Hadoop" (Rogers. 2019) covers this in detail. In this paper we only focus on the additions required for Kerberos constrained delegation.

It is important to note that the configuration of SAS Logon Manager for Kerberos must be applied to all services. Specifically, this means that the settings for sas.logon.kerberos must be defined as Global, as shown in Figure 8. This is the default setting with SAS Viya 3.5.

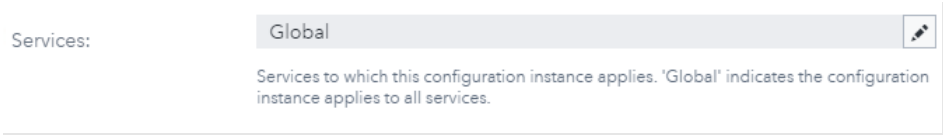

Figure 8. Applying sas.logon.kerberos Settings Globally

SAS OBJECT SPAWNER FOR SAS ENTERPRISE GUIDE

If you want to enable Kerberos for direct connections from SAS Enterprise Guide 8.2 to SAS Viya Object Spawner, you will also require additional Kerberos prerequisites. For SAS Viya 3.5 running on Linux you will require a service account with the SAS service principal name (SPN) registered against it. The format of the SPN will be SAS/<fully.qualified.hostname>, where the hostname needs to be the hostname of each host where SAS Viya Object Spawner is running.

You will also require a Kerberos keytab file for this service account as well. This Kerberos keytab file will need to be made available on all hosts where SAS Viya Object Spawner is running. Only the SAS installation account, sas, should have access to this Kerberos keytab file.

For SAS Viya 3.5 running on Windows nothing additional is required as SAS Viya Object Spawner runs as the local system account and the SPN will be registered automatically against the computer object.

ACTIVE DIRECTORY DELEGATION SETTINGS

In addition to the standard Kerberos prerequisites, additional settings are required for Kerberos constrained delegation within Active Directory. With "traditional" constrained delegation, the settings can be changed either using the Active Directory Users and Computers GUI tool or with Windows PowerShell command-lets. Resource-based constrained delegation is only supported when SAS Viya is deployed on Windows, can only be set using Windows PowerShell command-lets.

To configure "traditional" constrained delegation, the three service accounts must be updated. For SAS Viya 3.5, the HTTP principal must be able to delegate to the CAS principal and the SAS Launcher principal. The SAS Launcher principal must be able to delegate to the CAS principal and the data sources. The CAS principal must be able to delegate to the HTTP principal and the data sources. If you choose to enable direct connections from SAS Enterprise Guide 8.2 to SAS Object Spawner, the SAS principal must be able to delegate to the CAS principal and the data sources, as summarized in Table 1.

Application Principals	**Trusted to Delegate To…**
SAS Viya Web Applications (HTTP)	CAS (sascas) SAS Launcher (sas-launcher)
SAS Launcher (sas-launcher)	CAS (sascas) Data Sources
CAS (sascas)	SAS Viya Web Applications (HTTP) Data Sources
SAS Object Spawner (SAS) *Only required for direct SAS Enterprise Guide 8.2 connections.*	CAS (sascas) Data Sources

Table 1. "Traditional" Constrained Delegation Settings

To configure resource-based constrained delegation, the target or back-end services accounts must be updated. For SAS Viya Web Applications, the HTTP principal must allow delegation from the CAS principal. The SAS Launcher principal must allow delegation from the HTTP principal. The CAS principal must allow delegation from the HTTP and SAS Launcher principals, as well as the SAS principal for direct connections from SAS Enterprise Guide 8.2 to SAS Object Spawner. Data sources must allow delegation from the SAS Launcher principal, CAS principal, and SAS principal for the direct connections from SAS Enterprise Guide 8.2 to SAS Object Spawner, as summarized in Table 2.

Back-End Principals	**Trusted to Delegate From…**
SAS Viya Web Applications (HTTP)	CAS (sascas)
SAS Launcher (sas-launcher)	SAS Viya Web Applications (HTTP)
CAS (sascas)	SAS Viya Web Applications (HTTP) SAS Launcher (sas-launcher) SAS Object Spawner (SAS) *Only required for direct SAS Enterprise Guide 8.2 connections.*
Data Sources	SAS Launcher (sas-launcher) CAS (sascas) SAS Object Spawner (SAS) *Only required for direct SAS Enterprise Guide 8.2 connections.*

Table 2. Resource-Based Constrained Delegation Settings

Delegation from CAS to SAS Logon Manager is required to enable direct connections from SAS 9.4 to CAS. In this use-case as part of launching the CAS session, the end user must authenticate to SAS Logon Manager in SAS Viya 3.5. If this authentication to SAS Logon Manager in SAS Viya 3.5 cannot take place, then the CAS session launch will fail.

To configure "traditional" constrained delegation using the Active Directory Users and Computers GUI Tool, complete the following step for a given account:

1. Select the Delegation tab.
2. Select Trust this user for delegation to specified services only.
3. Select Use any authentication protocol (S4U2self).
4. Select **Add…**.
5. Search for users or computers with target SPN registered.

6. Select the target SPNs and select OK twice.

The settings on the Delegation tab will look like the example shown in Figure 9.

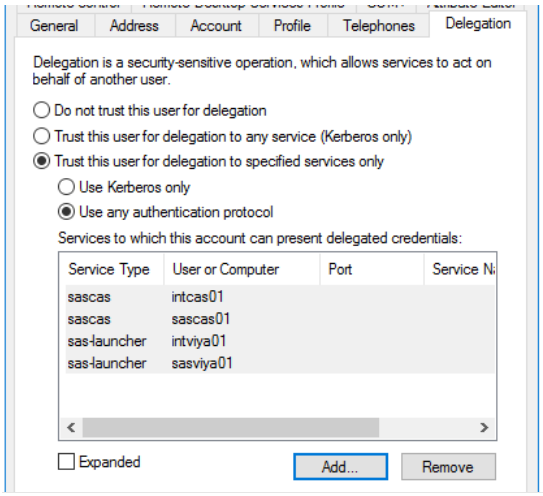

Figure 9. Example "Traditional" Constrained Delegation Settings

Alternatively, Windows PowerShell command-lets can be used to make the same changes. We will present an example for the HTTP principal service account. This will make the same changes we have just shown being made in the Active Directory Users and Computers GUI Tool. The same process can be repeated for the other services.

First, select the account object:

```
$Acct = Get-ADUser sasviya-HTTP
```

Then, set the back-end SPNs to which to delegate, in this case the CAS and SAS Launcher principals:

```
$Acct | Set-ADObject -Add @{'msDS-
AllowedToDelegateTo'=@('sascas/cascontroller.customer.com', 'sas-
launcher/sascompute.customer.com')}
```

Finally, enable protocol transition S4U2self:

```
$Acct | Set-ADAccountControl -TrustedToAuthForDelegation $true
```

You can confirm the changes you have made with the following:

```
Get-ADUser sasviya-HTTP -properties 'msDS-
AllowedToDelegateTo','TrustedToAuthForDelegation'
```

Using Windows PowerShell command-lets allows you to easily script updates. It will also be faster if you have many services to add to the AllowedToDelegateTo property compared to manually searching for services in the GUI.

Resource-based constrained delegation can only be configured using Windows PowerShell command-lets. Here we will present two examples; one with a single entry and another with multiple entries. The first example is for the HTTP principal service account where only a single service can delegate to it:

```
$user1 = Get-ADUser -Identity sasviya-sascas
Set-ADUser sasviya-HTTP –PrincipalsAllowedToDelegateToAccount $user1
```

The second example is for CAS where two services can delegate to it:

```
$user1 = Get-ADUser -Identity sasviya-HTTP
$user2 = Get-ADUser -Identity sasviya-launcher
Set-ADUser sasviya-sascas –PrincipalsAllowedToDelegateToAccount
@($user1,$user2)
```

This completes the Kerberos constrained delegation settings within Active Directory.

ADDITIONAL SERVICE ACCOUNT REQUIREMENTS

To enable S4U2self with Active Directory, the service must be able to authenticate itself. Only a user principal name (UPN) can authenticate using Kerberos to Active Directory. Therefore, the recommended approach is to set UPN = SPN for services that will perform S4U2self request. In the case of SAS Viya 3.5 this means the HTTP principal service account should have the UPN = SPN.

Alternatively, for the HTTP principal service account if your IT Team will not allow UPN = SPN then additional configuration is required for SAS Logon Manager. Also, the Kerberos keytab file must contain keys for both the UPN and SPN values. Remember the principal name is used as the salt when generating the long-term key based on the password for the account. So even though the password is the same, as it is the same account, the long-term keys will be different.

For services not using S4U2self requests we still recommend setting the UPN = SPN. If you need to have UPN ≠ SPN for CAS & SAS Launcher Server, then you will need the Kerberos keytab to either only contain the UPN or have the UPN listed first. Also, additional configuration for CAS and SAS Launcher Server will be required.

KERBEROS LIBRARY ON LINUX

SAS Viya on Linux requires the MIT Kerberos library. The minimum version required to support Kerberos constrained delegation is 1.11.6. If your SAS Viya environment is running RHEL 7.x and there are no issues, the standard operating system packages will provide a supported version of the MIT Kerberos library. However, if your SAS Viya environment is using RHEL 6.x, this does not provide the minimum supported version. As such, you will need to download the required version direct from MIT.

The requirement to use the MIT Kerberos library does not prevent you from using Quest Authentication Services, now known as One Identity, to manage the Linux host. If Quest is required for Active Directory integration and keytab management, then SAS Viya 3.5 must be configured to use MIT Kerberos libraries 1.11.6 or later. Further details on this are available from SAS Technical Support.

CONFIGURATION OF CONSTRAINED DELEGATION

The additional configuration of your SAS Viya 3.5 environment to support Kerberos constrained delegation is very straightforward.

LINUX AND WINDOWS: WITH DEFAULT SERVICE PRINCIPAL NAMES

To configure Kerberos constrained delegation with SAS Viya 3.5 on Linux, complete the following:

1. Within SAS Environment Manager select Configuration ⇨ Definitions.
2. Select sas.logon.kerberos.
3. Set disableDelegationWarning to true.

4. Set impersonate to true.
5. Save Changes and restart SAS Logon Manager.

Setting impersonate to true without configuring the HTTP principal for S4U2self will prevent Kerberos delegation from operating.

WINDOWS – ONLY: WITH DEFAULT SERVICE PRINCIPAL NAMES

On Windows, in addition to the SAS Logon Manager configuration, which is detailed above, a system environment variable must be set. To set the system environment variable, complete the following as a local administrator:

1. Open Control Panel ⇨ System and Security ⇨ System.
2. Select Advanced system settings and click Environment variables....
3. Click New.. under System variables.
   a. Add the following variable name and value:
      - Variable name: SAS_CONSTRAINED_DELEG_ENABLED
      - Variable value: 1
   b. Click Ok when finished.
4. Restart SAS Launcher Server and CAS Controller.

CONFIGURATION OF CUSTOM PRINCIPAL NAMES

If you need to use either custom SPNs or the service accounts, do not have UPN = SPN because additional configuration is required. To configure SAS Logon Manager when the UPN ≠ SPN complete the following:

1. Within SAS Environment Manager select Configuration ⇨ Definitions.
2. Select sas.logon.kerberos.
3. Set servicePrincipal to the name of the principal in the keytab.
   - Example: viya-http@customer.com
4. Set spn to the HTTP service principal name (SPN).
   - Example: HTTP/sasviyahttp.customer.com
5. Restart SAS Logon Manager.

A custom SPN or UPN format principal name is only supported on Linux for CAS. To set the principal name for CAS:

1. Update the following file:
   `/opt/sas/viya/config/etc/cas/default/casconfig_usermods.lua`
2. Add env.CAS_SERVER_PRINCIPAL = 'customSPN'.
   - Example UPN:
     ```
     env.CAS_SERVER_PRINCIPAL = 'viya-cas@customer.com'
     ```
   - Example custom SPN:
     ```
     env.CAS_SERVER_PRINCIPAL = 'cas/cascontroller.customer.com'
     ```

3. Restart the CAS controller.

To use a UPN format Kerberos keytab, the SAS_SERVICE_PRINCIPAL property must be set for SAS Launcher Server in the SAS Configuration Server. The property is set using the SAS BootStrap Config CLI. The example below is shown for Linux, but the same steps can be taken on Windows. Complete the following steps:

1. Set the required environment variables.

```
export CERT_DIR=/opt/sas/viya/config/etc/SASSecurityCertificateFramework/;
export CONSUL_TOKEN=`cat $CERT_DIR/tokens/consul/default/client.token`;
export LAUNCHER_KEY=config/launcher-server/global/environment
source /opt/sas/viya/config/consul.conf
```

2. Set the property values.

```
/opt/sas/viya/home/bin/sas-bootstrap-config kv write
$LAUNCHER_KEY/SAS_SERVICE_PRINCIPAL <<UPN>>
```

   - Example UPN:

```
/opt/sas/viya/home/bin/sas-bootstrap-config kv write
$LAUNCHER_KEY/SAS_SERVICE_PRINCIPAL viya-launcher@customer.com
```

3. Restart SAS Launcher Server.

This completes the configuration of your SAS Viya 3.5 environment when UPN ≠ SPN.

SAS OBJECT SPAWNER FOR SAS ENTERPRISE GUIDE

SAS Object Spawner on SAS Viya 3.5 can be updated to enable Kerberos constrained delegation direct connections from SAS Enterprise Guide 8.2. Remember this is only supported when TLS is disabled for SAS Viya Object Spawner. So, first to disable TLS for the Object Spawner in a Linux full deployment, complete the following steps:

1. Edit the file:

```
/opt/sas/viya/config/etc/spawner/default/spawner_usermods.sh
```

2. Add the following line to the bottom of the file:

```
spawner_options="${spawner_options//-ssl*/}"
```

3. Save your changes and restart the object spawner, for example, on RHEL:

```
systemctl restart sas-viya-spawner-default
```

4. Confirm your changes by looking at the command-line for SAS Object Spawner:

```
ps -ef |grep objsp
```

5. You should see that the command no-longer includes any of the following options:

   - -sslpvtkeyloc
   - -sslpvtkeypassfile
   - -sslcertloc
   - -sslcalistloc

To configure SAS Object Spawner for Kerberos authentication, complete the following:

1. Edit the file:

```
/opt/sas/viya/config/etc/spawner/default/spawner_usermods.sh
```

2. Add the following lines before the existing USERMODS line:

```
CMD_OPTIONS=-sspi
export KRB5_KTNAME=/opt/sas/sas.keytab
```

3. Update the existing USERMODS line to the following:

```
USERMODS="$JREOPTIONS $CMD_OPTIONS"
```

4. Save your changes and restart the object spawner, for example, on RHEL:

```
systemctl restart sas-viya-spawner-default
```

On Linux there is no additional configuration required to specifically support Kerberos constrained delegation. So long as the principal is correctly setup in Active Directory then Kerberos constrained delegation will be used. On Windows the system environment variable SAS_CONSTRAINED_DELEG_ENABLED, covered above, is also required.

TROUBLESHOOTING CONSTRAINED DELEGATION

To be able to troubleshoot Kerberos constrained delegation we need to follow the authentication flow and confirm what is happening at each stage. We need to look at the following:

1. The Client ticket cache
2. SAS Logon Manager
3. SAS Compute Server
4. SAS Cloud Analytic Server

When your end-users connect to the SAS Viya 3.5 web applications, they will be authenticated using Kerberos. The client will cache the HTTP service ticket, and this can be inspected using the Windows klist command. An example of is given in Output 1.

```
#1>     Client: end-user1 @ CUSTOMER.COM
        Server: HTTP/sasviya.customer.com @ CUSTOMER.COM
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
        Start Time: 1/17/2020 4:46:07 (local)
        End Time:   1/17/2020 14:45:46 (local)
        Renew Time: 1/24/2020 1:45:46 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called: dc1.customer.com
```

Output 1. Example Client Ticket Cache

Notice that the ticket flags do not include OK_AS_DELEGATE, which shows us that unconstrained delegation is not configured for the HTTP principal. Which means either we are using Kerberos constrained delegation, or no delegation has been configured for the account.

You have two choices for examining what is happening with SAS Logon Manager. First, you can dynamically enable logging for com.sas.logon.authentication, which will provide details of the authentication events occurring. This will provide details of both the external authentication using Kerberos, as shown in Output 2, as well as the internal group look-up from the Identities microservice.

To dynamically enable this logging information, complete the following:

1. In SAS Environment Manager select Configuration ⇨ Definitions ⇨ logging.level.
2. Select New Configuration.
3. For Services select SAS Logon Manager.
4. For level select DEBUG.
5. For name enter com.sas.logon.authentication.

Remember after collecting the information you require to set the level to WARN.

```
2020-01-17 05:24:05.070 DEBUG 7499 --- [1-auto-1-exec-8]
c.s.l.a.m.WebAuthenticationManager       : service [30eb5616cf0f0507]
Starting external authentication
for:org.springframework.security.kerberos.authentication.KerberosServiceReq
uestToken@6163d75a: Principal: com.sas.logon.user.ExternalUser@ff2c9643:
Username: end-user1; Password: [PROTECTED]; Enabled: true;
AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked:
true; Not granted any authorities; Credentials: [PROTECTED]; Authenticated:
true; Details:
org.springframework.security.web.authentication.WebAuthenticationDetails@ff
ff4c9c: RemoteIpAddress: 10.96.7.3; SessionId: 6f0ff16e-baa2-48ee-99e3-
13614e2310aa; Not granted any authorities
```

Output 2. Example com.sas.logon.authentication Log Message

Alternatively, with SAS Logon Manager to view the low-level Kerberos debug messages, you can complete the following:

1. In SAS Environment Manager select Configuration ⇨ All services ⇨ SAS Logon Manager.
2. Select Edit for the jvm properties.
3. Select Add property, and enter the following:
   a. Name = java_option_debug1
   b. Value = -Dsun.security.krb5.debug=true
4. Select Save twice and restart SAS Logon Manager.

This will provide the low-level information including details of accessing the Kerberos keytab and the principal obtained from the authentication. Output 3 illustrates the messages you will see for a successful Kerberos constrained delegation authentication. Notice the final message informs you that Kerberos constrained delegation has been used.

```
>>> EType: sun.security.krb5.internal.crypto.ArcFourHmacEType
Using builtin default etypes for permitted_enctypes
default etypes for permitted_enctypes: 18 17 16 23.
>>> EType: sun.security.krb5.internal.crypto.ArcFourHmacEType
MemoryCache: add 1579260742/000453/71E1457E408261686642ADFED91978D1/end-
user1@CUSTOMER.COM to end-
user1@CUSTOMER.COM|HTTP/sasviya.customer.com@CUSTOMER.COM
>>> KrbApReq: authenticate succeed.
Krb5Context setting peerSeqNumber to: 1150464000
>>> EType: sun.security.krb5.internal.crypto.ArcFourHmacEType
Krb5Context setting mySeqNumber to: 710375542
>>> Constrained deleg from GSSCaller{UNKNOWN}
```

Output 3. Example from -Dsun.security.krb5.debug=true Log Messages

For both SAS Compute Server and CAS, the first place to look for additional logging information is the microservice that launches the connection. For SAS Compute Server this would be the SAS Launcher Service, while for CAS this is most often the CAS Management Service. We can use additional logging on these services to see what credentials are used to authenticate the end user. To dynamically enable this logging information, complete the following:

1. In SAS Environment Manager select Configuration ⇨ Definitions ⇨ logging.level.
2. Select New Configuration.
3. For Services select either Launcher service or CAS Management service.
4. For level select DEBUG.
5. For name enter com.sas.credentials.jgss.

Remember after collecting the information you require to set the level back to WARN. This will log information to either <configuration directory>/var/log/launcher/default/ or <configuration directory>/var/log/cas-administration/default.

Output 4 shows the information that will be logged for SAS Launcher Service. We see the S4U2self request to obtain credentials for the end-user using impersonation. The HTTP principal is used for the S4U2self request. Then we see that the impersonated credentials for the end user are obtained and the remaining lifetime for these credentials.

```
2020-01-17 07:05:54.062 DEBUG 27414 --- [1-auto-1-exec-1]
c.s.c.jgss.GSSCredentialServiceCache     : end-user1 [c4331dc1bc967168]
Attempting to get credentials for end-user1@CUSTOMER.COM through
impersonation
Debug is  true storeKey true useTicketCache false useKeyTab true
doNotPrompt true ticketCache is null isInitiator true KeyTab is
///opt/sas/http.keytab refreshKrb5Config is false principal is
HTTP/sasviya.gellab.net tryFirstPass is false useFirstPass is false
storePass is false clearPass is false
principal is HTTP/sasviya.customer.com@CUSTOMER.COM
Will use keytab
Commit Succeeded

2020-01-17 07:05:54.071 DEBUG 27414 --- [1-auto-1-exec-1]
c.s.c.jgss.GSSCredentialServiceCache     : end-user1 [c4331dc1bc967168] Got
impersonated credentials for end-user1@CUSTOMER.COM with remaining lifetime
of 35999 secs
```

Output 4. Example Logging for SAS Launcher Service

On Linux, in addition to looking at the microservice that launches the connection, once the session has been launched the ticket cache can be inspected. The ticket cache will be generated in /tmp with the name tkt<rand>. This will be on either the host running SAS Compute Server or the CAS controller. The following command is run either as root or with sudo access, to view other users' files, will display the content of the ticket cache:

```
ls -tr /tmp/tkt*|tail -1 |xargs klist -c
```

Output 5 shows an example ticket cache for SAS Compute Server. You can see the principal name of the end user is given as the default principal. For Kerberos constrained delegation, it is important to confirm that the service principal for the first ticket matches the client for the krbtgt or TGT. This shows that the ticket is obtained through Kerberos constrained delegation.

```
Ticket cache: FILE:/tmp/tktfo85Jf
Default principal: end-user1@CUSTOMER.COM

Valid starting       Expires              Service principal
01/17/2020 07:05:54  01/17/2020 17:05:54  sas-
launcher/sasviya.customer.com@CUSTOMER.COM
01/17/2020 07:05:54  01/17/2020 17:05:54  krbtgt/CUSTOMER.COM@CUSTOMER.COM
        for client sas-launcher/sasviya.customer.com@CUSTOMER.COM, renew
until 01/24/2020 07:05:54
```

Output 5. Example SAS Compute Server Ticket Cache

Finally, the CAS controller log file can be checked, and the ticket cache created by the CAS controller verified. The CAS controller log file should show successful authentication with Kerberos and messages regarding the ticket cache, that CAS is unable to renew, as shown in Output 6.

```
2020-01-17T07:52:11,505 INFO  [00001289] MAIN NoUser MAIN [tkident.c:1323]
- User end-user1 successfully authenticated using the Kerberos
authentication provider.
2020-01-17T07:52:11,720 WARN  [00000007] 14651 end-user1 186
[tkidentgss.c:257] - Kerberos failure in function krb5_get_credentials:
Matching credential not found (filename: /tmp/tktG0QL4f) (96C73A8D).
2020-01-17T07:52:11,720 WARN  [00000007] 14651 end-user1 186
[tkidentgss.c:2570] - Kerberos ticket in cache /tmp/tktG0QL4f cannot be
renewed.
```

Output 6. Example CAS Controller Log Messages

Examining these different logs and the generated ticket caches will allow you to trace the authentication through the environment. You will then be able to spot the point in the flow where issues are occurring.

CONCLUSION

This paper has presented both the fundamentals of Kerberos constrained delegation and how SAS Viya 3.5 can be configured to use Kerberos constrained delegation. We covered several use-cases for how you can use constrained delegation in your SAS Viya 3.5 environment. We have addressed the current situation regarding the support of Kerberos constrained delegation with different data sources to make it clearer whether this will work in your environment.

We focused on the additional prerequisites and configuration required to enable Kerberos constrained delegation. We have presented the different ways you can correctly configure the different constrained delegation options in Active Directory. You've seen how simple the additional configuration within SAS Viya 3.5 is after you have configured normal Kerberos authentication.

Finally, we have provided you with some ways to troubleshoot and gather information on your environment. I hope that you find this paper a valuable resource. Additional documentation is provided in the following references.

REFERENCES

Microsoft. "[MS-SFU]: Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol." Available https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu/. Last modified September 9, 2018. Accessed on January 13, 2020.

Microsoft. "Windows 10 Device Guard and Credential Guard Demystified." Available https://docs.microsoft.com/en-gb/archive/blogs/ash/windows-10-device-guard-and-credential-guard-demystified. Last modified February 3, 2016. Accessed on January 13, 2020.

Rogers, Stuart J. 2017. "Kerberos Cross-Realm Authentication: Unraveling the Mysteries." *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. Available https://support.sas.com/resources/papers/proceedings17/SAS0623-2017.pdf.

Rogers, Stuart J. 2019. "Extending the Reach of Kerberos Authentication from SAS® Viya® 3.4 to Apache Hadoop." *Proceedings of the SAS Global Forum 2019 Conference*. Cary, NC: SAS Institute Inc. Available https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3000-2019.pdf.

ACKNOWLEDGMENTS

I would like to thank the following people for taking the time to review and contribute to this paper:

- Chuck Hunley
- Larry Noe
- Scott Parrish
- Mike Roda
- Younes Sammour

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Stuart J Rogers
SAS Institute Inc.
stuart.rogers@sas.com