Perforce Single Sign-On via Microsoft Active Directory

This document describes a method for implementing single sign-on (SSO) authentication for a Perforce server, using Microsoft Active Directory as the authentication provider. The scripts provided will not work on a Linux Kerberos server, but a Linux Perforce server will will work. These are sample scripts only.

Background

Perforce has offered an undocumented SSO mechanism for several years. As of the 2011.1 release, Single Sign-on is fully supported.

Essentially, Perforce's SSO mechanism relies on a client-side program to obtain a credential, and a server-side trigger to verify that credential. When the SSO mechanism is in place, a client must run the p4 login command, but if previously granted a Kerberos ticket, will not be prompted for a password.

Architecture

The basic architecture of this solution is shown in Figure 1.
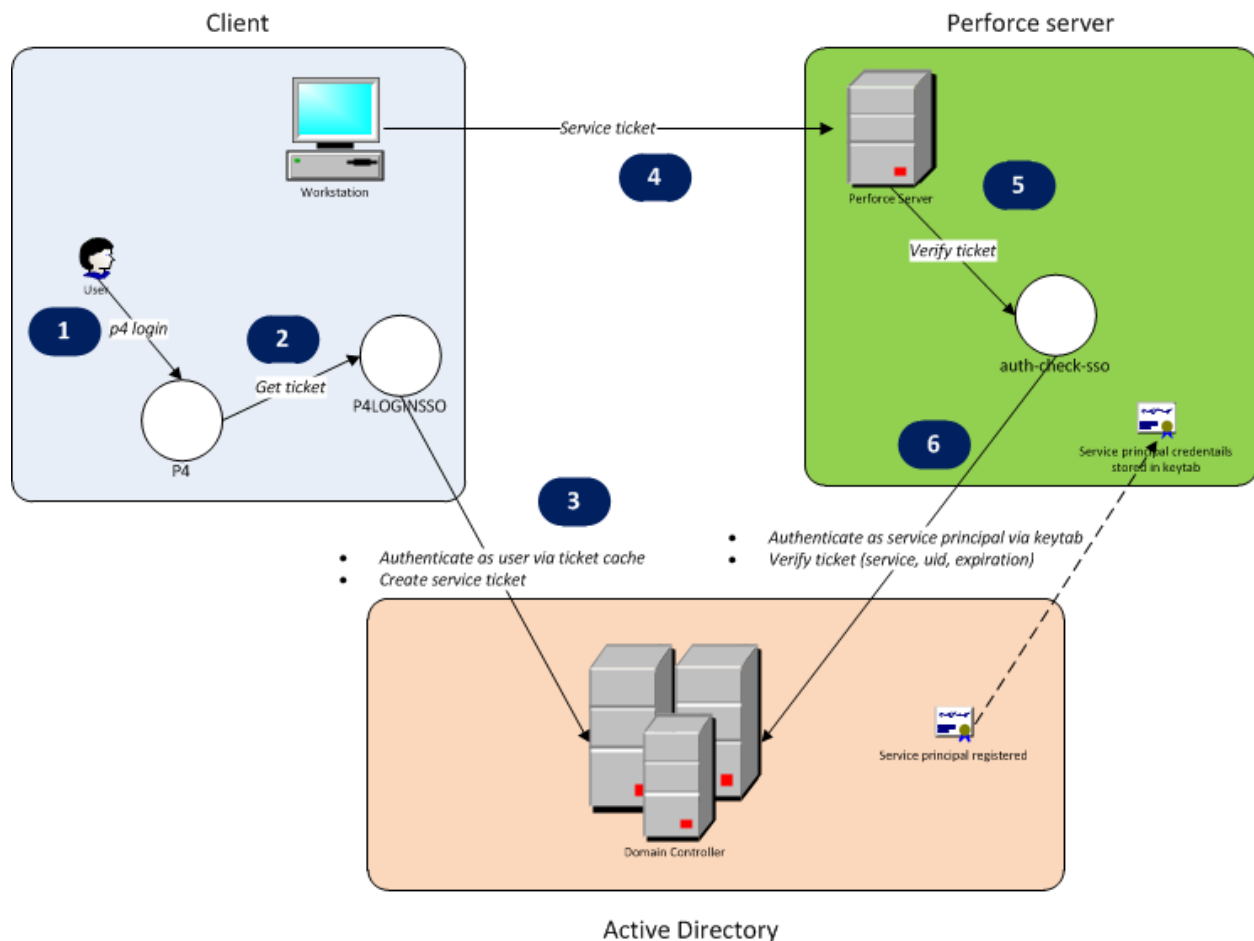
Figure 1: SSO Implementation architecture

The solution relies on the Kerberos protocol that is part of Active Directory. When this solution is used, a Perforce login request goes through several steps, as shown in Figure 1.

1. The user runs the *p4 login* command, either directly through the *p4* command line client, or through another client program.
2. The client program recognizes the presence of the *P4LOGINSSO* environment variable, and calls the program referenced by *P4LOGINSSO* to obtain a Kerberos ticket-granting ticket.

3. The *P4LOGINSSO* program authenticates to Active Directory as the current user by accessing the Kerberos ticket cache. It then creates a ticket-granting ticket for the Kerberos service used by Perforce.
4. The ticket is returned to the client program and passed to the Perforce server.
5. The Perforce server executes the *auth-check-sso* trigger to verify the ticket.
6. The *auth-check-sso* trigger authenticates itself as the Perforce service principal to Active Directory. It then verifies that the user's ticket is valid.

As the diagram indicates, the Perforce service principal's credentials must be stored on the Perforce server machine in the *keytab* file.

Implementation

This solution uses Java programs to retrieve and verify a Kerberos service ticket for the Perforce service. The using the techniques described at the former website http://www.javaactivedirectory.com. The source code was available, along with an Ant script for building and packaging. To currently obtain this code, download sso-p4 in the Perforce Workshop under Files → main → Download .zip.

Deploying the solution

The details of deploying this solution will almost certainly vary depending on your environment. Therefore, the instructions below are for reference only.

These instructions reference files described in Appendix 2: File manifest.

Test environment details

These instructions worked in a testing environment consisting of:

- A 32-bit Windows Server 2003 machine serving as the Active Directory domain controller (see Appendix 1: Domain Controller setup notes).
- A 2010.2 Perforce server running on the Windows Server 2003 machine. It also works with
- One client workstation running 32-bit Windows XP and the Perforce 2010.2 command line client.
- One client workstation running 64-bit Ubuntu 11 and the Perforce 2010.2 command line client. This workstation used *winbind* to join the Active Directory domain, as described in https://help.ubuntu.com/community/ActiveDirectoryWinbindHowto.

The following client programs were tested:

- The Perforce command line client, *p4*, version 2010.2
- P4V, version 2010.2
- P4Eclipse, version 2010.1
- P4SCC, version 2010.2

Of these four clients, p4*,* P4V, and P4Eclipse let a user authenticate without any password prompt. P4SCC gives a password prompt but authenticates successfully if the password field is left empty.

Note that SSO has also been successfully deployed on a newer Windows Active Directory server and Perforce server 2016.1.

Client

1. If using Windows, make sure that Windows will allow use of ticket-granting tickets.
2. On each client workstation, put the files *login.conf*, *sso-client.jar*, and *sso-client.bat/sso-client.sh* into a directory accessible to the user.

3. Modify the client's *sso-client.sh* or *sso-client.bat* file.

4. Modify the client's *login.conf* file, keeping only the *SignedOnUserLoginContext* section.

```
SignedOnUserLoginContext
{
      com.sun.security.auth.module.Krb5LoginModule required useTicketCache=true
doNotPrompt=true;
};
```

5. Customize the P4LOGINSSO driver script (*sso-client.bat* for Windows, *sso-client.sh* for non-Windows). The general format is:

"<path to JDK 6 java>" -jar <path to sso-client.jar> <path to SSO log file> <overwrite SSO log file during each run? true/false> <domain/realm> <domain controller> <path to login.conf> "<fully qualified service principal>"

Set the P4LOGINSSO environment variable to point to *sso-client.bat* or *sso-client.sh*

For example, on a Linux client, run

```
export P4LOGINSSO=/home/DEV/perforce/p4/sso-client.sh
```

Server

1. Make sure that Windows Active Directory server will allow use of ticket-granting tickets (TGT).

2. On the Perforce server machine, put the files *login.conf*, *sso-server.jar*, and *sso-client.bat/sso-client.sh* into a directory accessible to the account running the Perforce server. The "p4 triggers" configuration will access these files.

3. Customize the *ServicePrincipalLoginContext* section of *login.conf*. The general format is:

> com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true doNotPrompt=true
> principal=*"<fully qualified service principal>"* useKeyTab=true doNotPrompt=true storeKey=true
> keyTab="*<path to keytab>*"

Example server *login.conf* file. You will generate a keytab file location when you run ktpass or ktinit.

> ServicePrincipalLoginContext
> {
> com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true doNotPrompt=true
> principal="p4/bruno.ad.foo.com@AD.FOO.COM"
> useKeyTab=true doNotPrompt=true storeKey=true keyTab="**/etc/p4.keytab** debug=true";
> };

See Appendix 1: Domain Controller for a description of how to generate the *keytab* file. Refer to the documentation on Krb5LoginModule.

4. Customize the *sso-server.bat* driver script. The general format is:

"<path to JDK 6 java>" -jar <path to sso-client.jar> <path to SSO log file> <overwrite SSO log file during each run? true|false> <Perforce user ID passed as trigger argument> <domain/realm> <domain controller> <path to login.conf> "<fully qualified service principal>"

5. Install the trigger on the Perforce server:

   o Run

p4 triggers

   o Add an entry similar to:
     Triggers:
   > *sso auth-check-sso auth "<path to sso-server.bat> %user%"*

   o Restart the Perforce server.
     > p4 admin restart

Usage

Once the SSO solution is in place, a Perforce user will not have to supply a password when logging in, although command line users will first need to run *p4 login*.

Troubleshooting

Review the log files specified in the *sso-client.bat/sso-client.sh* file. If the client issues a ticket, then focus on fixing the server side.

Example client log:

Set Kerberos properties

Authenticated as current user

Authenticating as: bruno@AD.FOO.COM

Created client credentials

Created service context

Created service ticket

Got encoded service ticket

Service ticket:
YIIIeQYJKoZIhvcSAQICAQBugghoMIIIZKADAgEFoQMCAQ6iBwMFACAAAACjggeMYYIHiDCCB4SgAwIBBaESGxBBRC5NQVRIV09SS1MuQ09Noi0wK6ADA
gEAoSQwIhsCcDQbHGthbGlkLTAwLWFoLmFkLm1hdGh3b3Jrcy5jb22jggc4MIIHNKADAgEXoQMCAQKiggcmBIIHIrDC4HkvbGplQEinkVv+gaTWsFNjLy
MgaJo4xrTbxmcEhQYs/NDGeYprTrLbo5A4FggvCnpQK/Rh7L4s2mvx8IyaUU35JpRNW7eg8cFzBOUEWFlx863rvVFigMa+CJGWbf/vgUIK2L4H17/C1p8
lYPiTyM9pKBgwMZCpFacqRYbIUtb+u4KD6pyoISJk/oPa8Su/hvmjLpSIcxbc5doc5Ea8B1jG7g7Od/bU+195WLtk//lpZvuz0lVE2dgwppCys6lyzDhm
mOFbZEkDo48PooGdgYx1iYRYHXoYwYfCYjR/1cb9WjqI+nByIOPhy+aAwpViywyYxdQJ2kaF4KrckmWHAd0EqOV8B64sKzcad27mBEdfZBSlNVXXBUmPG
MdQqUNh+FYCi3bUNlhvkFMJlqfdslLyyaiOboH6NQldNt97ptqOqj+dXZyTpXoaaVH6qx8yt+e4NBJBBje55hA+aVLXs5qB+WDey3QZhejb6NmJsm6ml9
lCG1EbygJir2UD0uN6X5R/hgR7FelCGE8D83vL60szz1gdWF5R4GGq/IG7CyGnbgfDaaLZvoxQYayWeJvirbvIdTet51fpCEDpukKI26i5Qcl+00DLIjY
uEeZKQZkfw45HJo88bI/UaRH+KBPBbisF2Co5CXVvzk33FXdavZNeS2jcbidHdqZY4LPzzcO2BtGC68KmdF31rekahgxcBmJVjhvEvPp6dZGxXo9sEgmp
krFo06Cgs4vEXZnXAoX5fjI9THWIuzrzOKHyXjP8nRXRQvR6gmtZLUDp9yu2tGm1+dPW96K4J5AXY4TpuEjSVPZETiM4knY2SlmuqddERsrbhmk5vKo6J
eyQZfAxABSh+A0oduKKC325prWkTvZgTGwipds+IbTlelOJ2xRn5z2oMmTfwo3XurnfuAONFYDoXWymtvgBBc1K4v+EIc3XvPMPWHM5mrwHlVLdS27Kxg
hiEgqfRXId3nRXEMebBZoPULka6mFED40IJosgjNrQNd29YuQkTJ5Kh928yBn4ktPctnO0zqgZV78XMxyS5MnaknleozzxUjSAC+7f7LQPq/GSxdFsgMc
jwlOmZfmKArRVGVX2q5n1si1xzHpoGh+L3lna/Hms7pc/ZieHUaqrVVxuAYa9DhJxF60KkTToD8DWT/XFe8gM/AYbTLJpfBYAUV2ydQ/E06z3EaRi+294
lbwUq848UH8Cp1gt1CMu/oTr17QWMVfUsSqTgwufH9aB+ukYRFnOHQ8mDnnUP3Mkr3i3g9zJvxYfY6YX6uZ8qGKVyCmiSVOvtqRpuaalVsGl4XlUak3Gt
IJahJglR6nCAib56fsYdOk0sVnO2/KKWwKqUBBw751RP4BARed5mbV/aD+H/vVVXgOQVvNLDMdYbpetkSaiuwYdSOZqv5KuuxEza53ByJUMsf5tzVxNr9
tKuieDMOOKzs7u2UugMH7qwzaQbLwNSua8XRYwVJidgKl8EZbHxkZEHook3UinuYdyim4n36f2hKjG/jOczhtxGMoPM4NDdYdaRntbar7jMYotTnBDT8L
dwWPofplgxfj+djf5W7r9dVm2aEpiz2sc39FBquwmq91V6wECRVtOX1OPVAHYEGQaFGiOe1sicGxSZKqi7KB8zpcEtIgtxWpuBt+rfgTf9vK2/G0YQVAf
RHGIcWilKO+NAh5k/RWidsVz6k2pi+xA2qKwKbmhy+BXm5rUEbrmw3UcW5T5FxlLG0H1L4wMEh5e8Aaty1Iuc8ZkC8Q9/KLX64mCUACnZqeMvYBdPJtjB
Tqtd5TfRMOWmcOZcCZ+UUf/fUdqJCPUgsTKox9DRpOZH4Ovuh6BWx4oHuegrL2MovDDSx7htjLKQKkjupN2aaLNc7r4PeTVuHc9jaQCxrI7vHfeMgJWh1
DX9u5AiaRlOxJNzE5S5xItRtvsQBoQgJBnCRG1YXRYr2n4F7p54mBRtmfw2971I3eLd9UX798krvGrs4TC7oZo51hTtrvizU2YkHhZp0a9j9zfr590oXZ
ublwzgO923aRDpbiZJQl+PxIm1v0Px3OyWCW5LUcJRO7huKNa+vpsCJXaRNY7peT8Qs9XB271AfdSMNd2k0bg8DyG3vi2ohd7wEgwoiOmaSPa5S7tXM1K
yeNc8gntapAZcgYXsiAXOMnzFQG//qXVDujmbWNsXY113aL16WOGARcRmCorB+R9yiSyI0bbnOc+lL+SXGOMHWe2E0J2qTspiSbfCEFiK2r6ZgLK+bQcj
7RJQAH/MEFDjIdlc/4PTQtynvDS97BXWeDfAb08UjXubvI+A/bZ0i0m05jU3BLdqRHe+o0xYSibSMRaHtvHLikoM9idnLHaNXF1c8QorXgHxpYCsrge4J
78SKtQ0pssYy3Ek+bDZUSDoRgTC7s+7kT3nMFMIUYptqPRM6HIGbkOYKB/fkdJ4ri2pIG+MIG7oAMCAReigbMEgbDoYgAtj0ayt7RpR4yae8jxgAxHxrB
HfphwHNSCUoJYNrvoQ2fmmdj4KfH9CTAuui7Qv02XN/oyVqPOyoAr7eCZytQI7lB4ONoS1bFgPGBOhHNePI8bCPZdU8OuwWo/hpB+Ew/QhkdFbY+mWLvd
U1F9k+dSmK21XwxNbGsn6AbeJaoF6Pa45GT4OFEyjpJFvH2gUH2jcaSwdej6LZ719JfisShLiIJV6Un8nmjgM4ny/g==

Wrote ticket

Try various Windows server options.

Appendix 1: Domain Controller setup notes

This section is not a comprehensive description of how to configure an Active Directory domain controller. Rather, it describes particular steps that may be of interest. Use a Windows Kerberos/AD server, not a Linux Kerberos server.

As a general approach, the steps described at at http://d3planet.com/rtfb/2009/08/26/building-an-active-directory-domain-controller-for-development-using-vmware-workstation-pt-2/ were used to configure the domain controller for a test environment.

- The domain controller required a recent version of Windows Support Tools. For Windows Server 2003, the Support Tools are available from http://www.microsoft.com/downloads/details.aspx?FamilyID=96a35011-fd83-419d-939b-9a772ea2df90.
- The domain controller required a patch for a Kerberos encryption bug (Microsoft KB#833708).
- When configuring the service principal account, you may or may not have to uncheck the *Use DES encryption* option after creating the account. If you have problems, change the password after this option is selected.
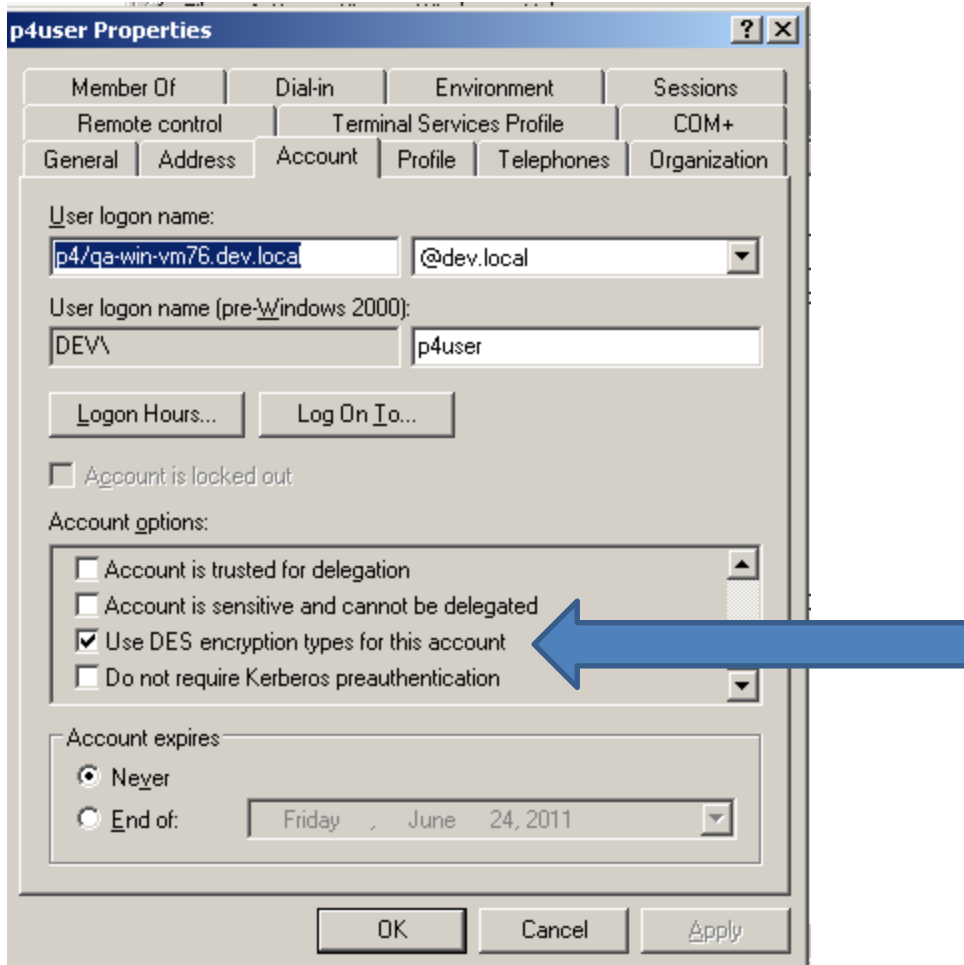
Figure 2: DES encryption option

- When registering the service principal account on the Perforce server machine, create the *keytab* file using *ktpass*. For example, in a testing environment, using *p4user* as the service principal account for the *p4* service, in a domain called *DEV.LOCAL*:

  > *ktpass -princ p4/domain-controller.dev.local@DEV.LOCAL -ptype KRB5_NT_PRINCIPAL -crypto DES-CBC-CRC -mapuser "p4user" -pass * -out c:\keytab -kvno 0*

  Alternatively, you can run on the Perforce server

  > *ktinit -kt /etc/p4.keytab*

  where */etc/p4.keytab* is specified in the login.conf file on the Perforce server. Note that DES-CBC-CRC is used as an example; others can be used as well.

  The *keytab* file is then used on the Perforce server, and is referenced in *login.conf*.

- Link the *p4user* account to the *p4* service using *setspn*:

setspn -a p4/domain-controller.dev.local@DEV.LOCAL p4user

Appendix 2: File manifest

Apache-license.txt	**Apache Software Foundation license**
Java\KerbClient	Java source code and JAR file for client side program
Java\KerbServer	Java source code and JAR file for server side trigger
Java\login.conf	*Login.conf* file used on both client and server
Java\sso-client.bat	Batch file used to run the client Java program as the *P4LOGINSSO* program on Windows
Java\sso-client.sh	Shell script used to run the client Java program as the *P4LOGINSSO* program on non-Windows platforms
Java\sso-server.bat	Batch file used as *the auth-check-sso* trigger
Java\build.xml	Ant script used for building and packaging the Java code. Depends on log4j and commons-codec Apache libraries.

Appendix 3: FAQ

Question	Answer
Why write this solution in Java?	Java has well-structured and documented Kerberos libraries. The same Java programs can be used on multiple platforms.
How much can Perforce support me if I have problems implementing this solution in my environment?	Perforce technical support can provide assistance with Perforce's SSO mechanism. Unofficial assistance may be available for the Java programs. Detailed assistance with Active Directory problems may be outside the scope of Perforce's ability to help.
What happens if a client doesn't set the *P4LOGINSSO* variable?	They will fall back to regular authentication. You may wish to provide an *auth-check* trigger so that they can still provide their Active Directory password.
What happens if Active Directory is down?	A client can unset *P4LOGINSSO* and fall back to regular password authentication. If you need to remove the *auth-check-sso* trigger on the server and are locked out of Perforce, consult http://kb.perforce.com/article/838/regaining-super-user-access-to-your-perforce-database and http://kb.perforce.com/article/834/p4d-xf-flag.