Active Directory – Single Sign on using 4D 2004

By Thomas Maul, General Manager, 4D Germany.
TN 06-02

Summary

A MS Windows domain environment allows single sign in, which means a user authenticates only once on the computer, then can use all network resources like file server, printers, mail servers or databases without the need to renter the password.

This also allows the administrator to have a centralized user management. Creating or removing users in this directory, or modifying the group affects the whole network.

This technical note shows how to integrate a 4D application without the need for a plug-in.

Benefit for your customers

Microsoft writes in "Active Directory Benefits for Mid-market.doc":
Active Directory is the integrated, distributed directory service that is included with Microsoft Windows Server 2003 and Microsoft Windows 2000 Server. Integrated with Active Directory are many of the applications and services that previously required a separate, distinct directory and user_id/password to be managed for each application or service. In Windows NT 4.0, for example, a directory was required for the domain itself, a separate directory for Exchange mailboxes and distribution lists, and separate directories for remote access, database, and other applications. In some cases, separate passwords were required for each application. With Active Directory, the administrator of the organization can add a user to Active Directory and through that single entry enable remote access to the network, enable the same user account for Exchange messaging, that same user for database access for accounting, client relationship management, or other applications. Not only is it possible to use Active Directory as a multi-purpose directory in this fashion but by doing so a company enables single sign-on for its users. Once a user logs in to Windows their Active Directory credential is the key that will automatically unlock all of the applications or services that they have been enabled for, including 3rd party applications that utilize Windows integrated authentication.
Source: http://www.microsoft.com/WindowsServer2003/techinfo/overview/adsmallbiz.mspx

Another benefit is increased security for your application. Some custom solutions request increased security that goes beyond the concept of User_ID/Password. Systems like Smartcards or Fingerprint scanners are more and more common these

days. An access system supported by Windows Active Directory will also be supported from your application – without additional work.

Active Directory

A MS Windows computer is either a stand alone machine with a local user list and security rights or a member of a domain, sharing the domain user list and with centralized group policies.

To test this example or to use this code you need to have a domain controller (usually Windows 2000 or 2003 Server or Small Business Server) and your Windows computer must be a member of this domain. With a standalone Windows computer you can try this example, but will be of course not able to receive group memberships from the domain controller.

Important: This code cannot be used on a Mac OS client.

Usage with 4D – Method AD_UserGroups

This technical note introduces a single 4D method which returns the following information:

- current authenticated user name
- current domain name
- current domain DNS suffix
- list of groups of this user

```
$out:=""

C_BLOB($result)

PLATFORM PROPERTIES($plat)
If ($plat#3)
   ALERT("This code uses Active Directory. It can be used on Windows only")
Else
   $path:=Temporary folder+"myscript.vbs"
   $status:=Test path name($path)
   If ($status=1)
      DELETE DOCUMENT($path)
   End if
   $ref:=Create document($path)
   $cr:=Char(13)+Char(10)
   $text:="Dim WshNetwork"+$cr
   $text:=$text+"Dim FSO"+$cr
   $text:=$text+"Dim strUserName"+$cr
   $text:=$text+"Dim strUserDomain"+$cr
   $text:=$text+"Dim strDNSDomain"+$cr
   $text:=$text+"Dim bjRootDSE"+$cr
```

```
   $text:=$text+"Set WshNetwork = WScript.CreateObject(\"WScript.Network\")"+$cr
   $text:=$text+"Set FSO = CreateObject(\"Scripting.FileSystemObject\")"+$cr
   $text:=$text+"strUserName = WSHNetwork.UserName"+$cr
   $text:=$text+"While strUserName = \"\""+$cr
   $text:=$text+"WScript.Sleep 100"+$cr
   $text:=$text+"strUserName = WSHNetwork.UserName"+$cr
   $text:=$text+"Wend"+$cr
   $text:=$text+"strUserDomain = WSHNetwork.UserDomain"+$cr
   $text:=$text+"WScript.StdOut.Write strUserName & vbCR & vbLF"+$cr
   $text:=$text+"WScript.StdOut.Write strUserDomain & vbCR & vbLF"+$cr
   $text:=$text+"Set objRootDSE = GetObject(\"LDAP://RootDSE\") "+$cr
   $text:=$text+"strDNSDomain = objRootDSE.Get(\"defaultNamingContext\")"+$cr
   $text:=$text+"WScript.StdOut.Write strDNSDomain & vbCR & vbLF"+$cr
   $text:=$text+"Set CreateMemberOfObject = CreateObject(\"Scripting.Dictionary\")"+$cr
   $text:=$text+"CreateMemberOfObject.CompareMode = vbTextCompare"+$cr
   $text:=$text+"Set objUser = GetObject(\"WinNT://\" & strUserDomain & \"/\" & strUserName & \",user\")"+$cr
   $text:=$text+"For Each objGroup In objUser.Groups"+$cr
   $text:=$text+"WScript.StdOut.Write objGroup.Name & vbCR & vbLF"+$cr
   $text:=$text+"Next"+$cr
   SEND PACKET($ref;$text)
   CLOSE DOCUMENT($ref)

   $out:=""
   $input:=""
   $err:=""
   $cmd:="cscript  //Nologo "+Char(34)+$path+Char(34)
   SET ENVIRONMENT VARIABLE("_4D_OPTION_HIDE_CONSOLE";"true")
   LAUNCH EXTERNAL PROCESS($cmd;$input;$out;$err)
   DELETE DOCUMENT($path)
   $out:=Replace string($out;Char(13)+Char(10);Char(13))
   $out:=Win to Mac($out)
   If ($err#"")
      $out:="Error: "+$err+Char(13)+Char(13)+$out
   End if

   If (Count parameters>0)
      Case of
         : (Type($1->)=Text array )
            ARRAY TEXT($1->;0)
            $pos:=Position(Char(13);$out)
            While ($pos>0)
               APPEND TO ARRAY($1->;Substring($out;1;$pos-1))
               $out:=Substring($out;$pos+1)
               $pos:=Position(Char(13);$out)
            End while
            If ($out#"")
               APPEND TO ARRAY($1->;$out)
            End if
         : (Type($1->)=Is Text )
            $1->:=$out
      End case
   End if
End if
```

This data can be returned either as text or as text array.

Example:

Thomas

DE4D
DC=de,DC=4D,DC=local
SBS Mobile Users
4DGmbH
Web Workplace Users
Domain Admins
Domain Power Users
Domain Users
Designer

- The first line is the name of the user, as entered in the Windows Login dialog and created from the admin in Active Directory.
- The second line is the name of the domain the user is signed on. If the user is not part of a domain, this line will contain the name of the computer.
- Third line is the DNS suffix of the domain, in the syntax needed for LDAP communication. In this example the suffix is "de.4D.local". If the user is not a member of a domain, this line will contain an error message and is the last line (or last element if an array is used).

The following lines/elements contain the name of all security groups the user belongs to. Note that it are security groups, other groups like Exchange groups are not returned.

The method can be called as:

```
ARRAY TEXT(arrText;0)
AD_UserGroups (->arrText)
```

Or

```
C_TEXT(vText)
AD_UserGroups (->vText)
```

The method expects a pointer to a text variable or a text array.

Using your own password system

In this case your existing code will open in the On Startup a dialog which allows the user to enter his user name and password. Before you do that, you use the method AD_UserGroups to check if the user is allowed into your system. If yes, you allow the login without showing your own password dialog. If not – you may either reject the login or open your own dialog, depending of the security guidelines of your customer.

The code can look like this:

```
$domain:=[Prefs]Domainname
$domainsuffix:=[Prefs]Domain_Suffix
$groupname:=[Prefs]Main_Group

PLATFORM PROPERTIES($plat)
If (($plat=3) & (Not(Shift down)))
```

```
      ` this gives us a chance to use the system on Mac
  ` or Windows computers not belonging to a domain server

ARRAY TEXT($users;0)
AD_UserGroups (->$users)
If (Size of array($users)>3)  ` else cannot be a legal login
   If (($users{2}=$domain) & ($users{3}=$domainsuffix))
      $pos:=Find in array($users;$groupname;4)
      If ($pos>0)
         ` User exist and is in the group - so let him in
```

If the user reaches this line, he belongs to our main group (a group allowing the usage of the database). Of course you may have other groups to allow to use different parts of your databases. The "Else" parts will vary based on your needs. You may open your own dialog or call QUIT 4D.

The method then resumes with the code below, but keep in mind that :

```
         ` User exist and is in the group - so let him in

      $helpblob:=[Prefs]Adminpassword
      DECRYPT BLOB($helpblob;[Prefs]Private_key;[Prefs]Public_key)
      $password:=BLOB to text($helpblob;Text without length )
      CHANGE CURRENT USER([Prefs]Adminuser;$password)  `  this gives us the right to create/modify users

       `check the current valid/assigned groups
      ARRAY TEXT($groupnames;0)
      ARRAY LONGINT($grouplist;0)
      ARRAY LONGINT($grouplistold;0)
      GET GROUP LIST($groupnames;$grouplist)
      For ($i;1;Size of array($groupnames))
         $pos:=Find in array($users;$groupnames{$i};4)
         If ($pos<1)
            $groupnames{$i}:=""
         End if
      End for
      $pos:=Find in array($groupnames;"")
      While ($pos>0)
         DELETE ELEMENT($groupnames;$pos)
         DELETE ELEMENT($grouplist;$pos)
         $pos:=Find in array($groupnames;"")
      End while

       `check if user exists in 4D password system, else create
      $Username:=$users{2}+"/"+$users{1}
      ARRAY TEXT($Userlist;0)
      ARRAY LONGINT($UserID;0)
      GET USER LIST($Userlist;$userID)
      For ($i;1;Size of array($userID))
         If (Is user deleted($UserID{$i}))
            $Userlist{$i}:="xxxxxxxxxxxx"
         End if
      End for
      $pos:=Find in array($Userlist;$Username)
      If ($pos<0)
         $password:=""
         For ($i;1;15)
            $password:=$password+Char((Random%(90-65+1))+65)
```

```
            End for
            $newID:=Set user properties(-2;$Username;"";$password;0;!00/00/00!;$grouplist)
            QUERY([Users];[Users]Name=$Username)
            If (Records in selection([Users])=0)
               CREATE RECORD([Users])
               [Users]Name:=$Username
               [Users]ID:=$newID
            End if
            TEXT TO BLOB($password;[Users]Password;Text without length )
            ENCRYPT BLOB([Users]Password;[Prefs]Private_key;[Prefs]Public_key)
            SAVE RECORD([Users])
         Else
            $newID:=$userID{$pos}
            QUERY([Users];[Users]Name=$Username)
            DECRYPT BLOB([Users]Password;[Prefs]Private_key;[Prefs]Public_key)
            $password:=BLOB to text([Users]Password;Text without length )
              ` check if groups are still valid
            GET USER PROPERTIES($newID;$name;$startup;$passdummy;$login;$logdate;$grouplistold)
            $modified:=False
            If (Size of array($grouplist)#Size of array($grouplistold))
               $modified:=True
            Else
               For ($i;1;Size of array($grouplistold))
                  If ($grouplistold{$i}#$grouplist{$i})
                     $modified:=True
                  End if
               End for
            End if
            If ($modified)
               $newID:=Set user properties($newID;$name;"";$password;$login;$logdate;$grouplist)
            End if
         End if

         CHANGE CURRENT USER("__login";"")
           ` login back, just to be sure that if login fails, we have no rights
         CHANGE CURRENT USER($Username;$password)
      End if
   End if
End if

If (($newID=0) & (Current user="__login"))  ` this is only the case on Mac or without domain
   CHANGE CURRENT USER
End if

If (Current user="__Login")
   ALERT("Sorry, there was a problem with login, please try again...")
   QUIT 4D
Else
    ` do your startup job here...
    ` Startup
End if
```

Using 4D's password system

Using 4D's password system has at least three major advantages. You can use the command "Current User" everywhere in your code, even in Triggers, to identify the

current user, 4D's log file (for 4D Backup) shows who has deleted your 1000 most important customers and you can use the command "Locked Attributes" to identify the user who is locking a record.

Using 4D's password system does not necessary mean you use 4D's password editor or login dialog. 4D 2004's modified command CHANGE CURRENT USER allows you to hide 4D's system totally, so the user never sees any part of it.

To integrate with Active Directory we can use 4D's system in an invisible way: automatically create new users, assign them to groups, and remove them from group and perform an invisible login.

This allows the Active Directory Administrator to add (or delete) users and assign them to groups depending of their work position. Your database automatically adapt to these modifications without the need for the admin to use 4D's password and group editor.

To do that, a little more code is needed. The example database contains a method named "Startup_Example" which shows a solution. To implement this solution in your database you assign to the design user a password only known to you for development. The admin user is set to a predefined password used only from your code, so the end user never needs to know about it. The admin user account is never used by a 'real' user. The 3rd predefined user does not have a password, this user account is used as "Default User", so 4D runs the On Startup method without displaying the login dialog. This special user has no access rights, he does not belong to any group. The user account is only used to run during the On Startup method.

The login steps are:

- check if the user is a domain member of the allowed domain
- check if the user is part of the database main group
- use CHANCE CURRENT USER to switch to 4D admin rights
- read all groups from 4D – check which groups the current user is a valid member of. Create an array of group membership
- check if the user exists in 4D's password system, if not, create him, assign a random calculated password, encrypt it and, save it inside a record
- check the group membership of the user, assign the current groups
- using Change Current User log in with the real user account, use the stored password (encrypt it automatically).
- Last step: check if the Current User is still the default user. Then something went wrong (illegal access), Quit 4D otherwise begin with On Startup.

About the method AD_UserGroups

The method uses a VBS script to read the network information's. The script creates a Windows Network object, and then read the current user and domain name from the

object. Using this information, it requests from the network the group memberships and finally uses LDAP to request the DNS Suffix of the root domain.
The result is returned to 4D as a text, which is either put directly into the 4D text variable or split into elements and returned as text array.