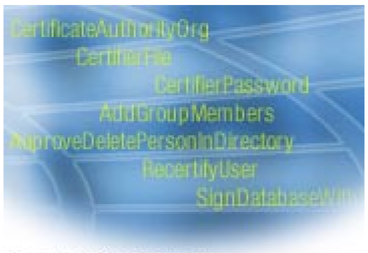
LDD Today

developerWorks

Lotus. software



LotusScript: The NotesAdministrationProcess Class in Notes/Domino 6

Level: Advanced

Works with: Notes/Domino 6

Updated: 01-May-2003

by Sally Blanning DeJean
and David DeJean

This is the fourth in a series of articles that take a close look at the new LotusScript classes and enhancements to the LotusScript language in Notes/Domino 6. Previous articles dealt with the new LotusScript classes for manipulating rich text elements ("**LotusScript: Rich text objects in Notes/Domino 6**") and LotusScript support for XML ("**LotusScript: XML classes in Notes/Domino 6**" and "**LotusScript: More XML classes in Notes/Domino 6**"). In this fourth article, we look at the new NotesAdministrationProcess class. As with the previous articles, this one assumes that you are an experienced Domino application developer with knowledge of the LotusScript programming language.

The NotesAdministrationProcess class

It isn't news when changes in a programming language make developers' lives easier—give them more tools or better debugging. But it is news when enhancements to a language wind up benefiting administrators. For that reason, the new NotesAdministrationProcess class in LotusScript for Notes/Domino 6 is news.

This new class brings Domino developers and administrators a little closer. It gives developers a chance to build a tool for administrators that they can use to securely delegate many routine administrative tasks such as recertifying users, adding users to groups, and deleting user accounts.

The new NotesAdministrationProcess class offers properties and methods that support tasks that in the past could be performed only from the Domino Administrator client or in the Domino Directory and were the exclusive domain of administrators.

In many organizations, Domino administration is separate from Domino application development, and developers and ordinary users have little or no access to administration tasks. Yet that separation has meant that Domino administrators have not been able to delegate some of the responsibilities they routinely perform. For example, to recertify a user, an administrator had to have access to a cert.id file, the certificate for the organization or organizational unit. For security reasons most administrators do not share this ID and its password with non-administrators.

Working outward from AdminP

Many administration tasks deal with people and databases. When an employee is terminated, the usual procedure is for an HR representative to send an email notification to the administrator notifying her that the employee should be removed from the Domino Directory. Using either the administration client or agents in the Directory, the administrator then marks the employee for deletion. At a later time, the Administration Process runs and removes the employee from the Directory and, depending on database and ACL settings, from Readers fields and ACLs.

The Administration Process, or AdminP, is a server task that was developed to be a kind of automated majordomo for the Domino Directory. It helped minimize the constant churn of changes as users are added, deleted, and modified while preserving the security of the change process and making the Domino administrator's gatekeeper role a little less burdensome. Rather than edit the Domino Directory directly, administrators create change requests using actions in the Domino Directory or tools in the Domino Administrator client. These requests go into a secure database named Administration Requests. The AdminP process checks this database for changes to the Domino Directory. It batches the requests for execution on a schedule designed to have the least impact on the system's performance—tasks such as certifying a user, adding a user to a group, removing a user, renaming a user, removing a mail file, deleting a database replica, deleting a server, recertifying a server, and many more.

Using the properties and methods of the NotesAdministrationProcess class, a developer can create a database that extends the benefits of AdminP and the Administration Requests database further outward into the organization to let non-administrators initiate many of these routine tasks of Notes administration. For example, developers who maintain databases can submit requests to create or move replicas. Help desk personnel can submit requests to recertify IDs about to expire. A Human Resources representative can submit name changes or account deletions.

Each request becomes a document, and agents in the database can run with greater authority than the requestors themselves have to add those requests to the Administration Requests database. The agent that acts on the document the HR representative creates could run as a scheduled agent on the server or on behalf of someone with a greater authority than that of the HR representative. Finally the AdminP server task, which runs on the server, acts on the requests entered in the Administration Requests database. As it processes requests, it marks their status in the Administration Requests, providing an audit trail of the actions taken.

The tasks that can be scripted with the NotesAdministrationProcess class include operations that delete, add, change, move, recertify, sign, and approve. You can add data to groups, Person documents, or clusters. You can approve different kinds of deletions, name changes, and moves. You can change names and HTTP passwords. You can search the domain for users, servers, and groups. You can recertify and sign databases.

Here are the properties and methods of the NotesAdministrationProcess class covered in this article. For a complete list of properties and methods, see the **NotesAdministrationProcess class properties and methods sidebar**

Properties

- CertificateAuthorityOrg
- CertifierFile
- CertifierPassword
- IsCertificateAuthorityAvailable

- UseCertificateAuthority

Methods

- AddGroupMembers
- ApproveDeletePersonInDirectory
- ChangeHTTPPassword
- DeleteUser
- FindGroupInDomain
- FindUserInDomain
- RecertifyUser
- RenameGroup
- RenameNotesUser
- SetUserPasswordSettings
- SignDatabaseWithServerID

This article concentrates on scripts that pertain mainly to users and databases. Methods such as DeleteServer, AddServerToCluster, and ApproveDeleteServerInDirectory are not addressed here. It is unlikely that they would be delegated to non-administrators.

Some things to do before you start

Before exploring the NotesAdministrationProcess class, it is important to review a checklist of tasks and databases that should be active. Domino administrators almost always manage these. Briefly, they are:

- The Administration Server must be in the Domino domain where the Domino Directory is located. This server runs the Administration Process task (AdminP) which makes changes to the Domino Directory and to ACLs and Readers, Authors, and Names fields of designated databases. It also manages replication events. In a large organization, there may be more than one Administration server.
- The Advanced tab on individual database ACL settings should have the Administration server selected, otherwise, the AdminP task will ignore those databases.
- The Administration Requests database (admin4.nsf) must reside on the Administration server and should replicate throughout a domain. The Administration Requests database tracks all the tasks performed by AdminP and displays their status. It can be a helpful diagnostic tool for administrators who check on the status of a task.
- The Certification Log database must also reside on the Administration server. This database keeps a record of certifications, recertifications, and name changes.
- The AdminP server task must be running on the Administration server and other servers in the domain. This task processes the requests submitted to the Administration Requests database.
- The CA task must be running on the Administration server for CA features to work. The Certificate Authority (CA) is a new feature of Notes/Domino 6. It provides a way of managing the Domino registration and certificate(s) on a server with greater security. The certificate can be secured on the

server without giving administrators direct access to it. The CA is not required. It is still possible to manage certificates with a certifier and password that administrators know and often share. (For more about the CA see IBM Redbook "**Upgrading to Lotus Notes and Domino 6**," and the *LDD Today* article, "**Be the authority on the Domino 6 Certificate Authority**.")

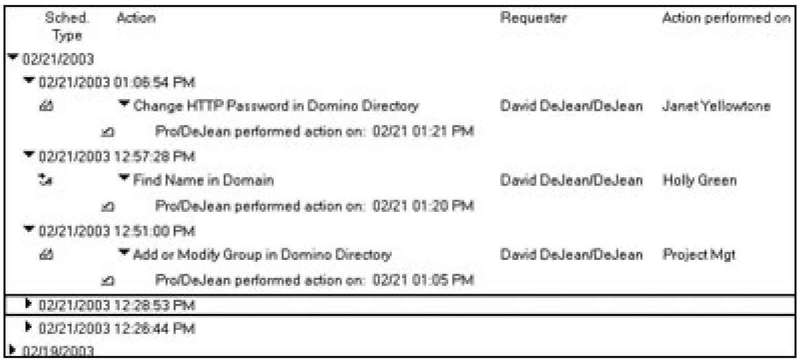
A database of the sample forms and scripts used in this article is available for download from the **Sandbox**. The database is called LS Admin Process. It contains forms in which users enter information that is then used by agents to create administration requests that are subsequently sent to the Administration Request database. Access to the LS Admin Process database is controlled through its ACL. To use the agents in this database, you have to modify them—changing references to servers, certifiers, or the CA and signing them in a manner appropriate to your environment. Also, the speed with which the changes occur depends on the frequency with which the AdminP process works. Your administrator can help you determine this.

Creating a script with the NotesAdministrationProcess class

The NotesAdministrationProcess class properties are attributes of the things a person works with. Examples include the CertifierFile property, representing a certifier ID file, and UseCertificateAuthority, representing the new Certificate Authority. CertificateExpiration is a property representing the expiration date of the certifier.

The class methods represent the actions of the administration process. For example, AddGroupMembers requests that persons and groups be added to a group document in the Domino Directory and DeleteGroup requests that the group be removed. DeleteUser begins a process to mark a Person document for removal, and ApproveMailFileDeletion adds a request to complete the pending request to also remove the person's mail file. SignDatabaseWithServerID performs the task of signing all design elements of a database with the Server's ID.

The sample view below from the Administration Requests database shows three requests: Change an HTTP Password, Find a Name, and Add or Modify a group, that have been processed. LotusScript agents using the NotesAdministrationProcess classes submitted all these requests.



Sched. Type	Action	Requester	Action performed on
▼ 02/21/2003			
▼ 02/21/2003 01:06:54 PM			
	▼ Change HTTP Password in Domino Directory	David DeJean/DeJean	Janet Yellowtone
	Pro/DeJean performed action on: 02/21 01:21 PM		
▼ 02/21/2003 12:57:28 PM			
	▼ Find Name in Domain	David DeJean/DeJean	Holly Green
	Pro/DeJean performed action on: 02/21 01:20 PM		
▼ 02/21/2003 12:51:00 PM			
	▼ Add or Modify Group in Domino Directory	David DeJean/DeJean	Project Mgt
	Pro/DeJean performed action on: 02/21 01:05 PM		
▶ 02/21/2003 12:28:53 PM			
▶ 02/21/2003 12:26:44 PM			
▶ 02/19/2003			

Creating a script with NotesAdministrationProcess is not complex. First declare a variable as the NotesAdministrationProcess(server$). Then instantiate it using the CreateAdministrationProcess, a method of the NotesSession class.

In the following agent example (named 01. Delete a group) the Domain/ServerName parameter is the domain and server where the Administration Requests database (admin4.nsf) is running. An empty string means the local computer. Remember, the ID running the agent must have the access privileges to the Domino Directory stated earlier for this agent to run. The agent adds a request to the Administration Request database to remove the group West Sales from the Domino Directory.

```
Dim session As New NotesSession
Dim adminp As NotesAdministrationProcess
Set adminp = session.CreateAdministrationProcess("Domain/ServerName")
Call adminp.DeleteGroup("West Sales", True)
```

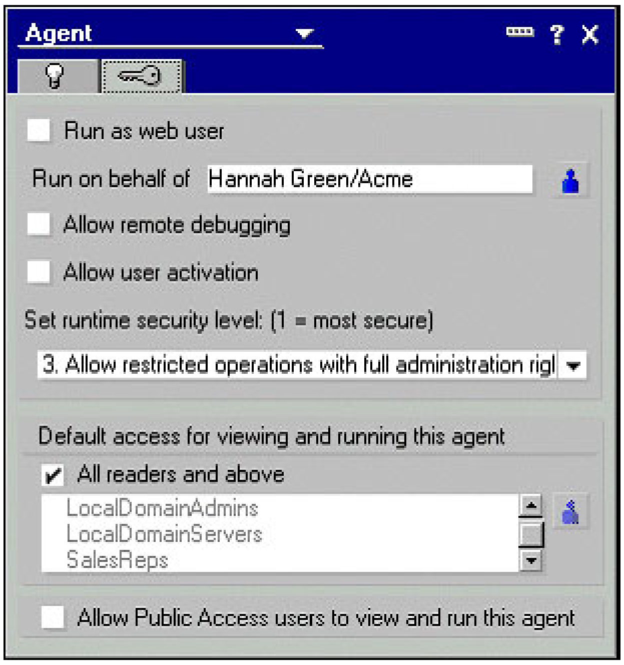
The agent is scheduled to run on the server, and it must be signed by an ID that has at least Editor access to the admin4.nsf database. The agent signer must have Author access with the Create Document privilege in the Domino Directory with the UserCreator role enabled. When the AdminP task runs, it performs the request.

Agent settings

Notes/Domino 6 provides a new capability, Run on behalf of, that lets users activate agents that they normally would not be allowed to run. When this setting is used as shown below, it allows someone to activate an agent and run it with all the rights and privileges of the person named. So, a person without rights to run LotusScript or who does not have Author access to the Administration Requests database can run the agent "on behalf of" someone who does have those rights and privileges.

You can further refine the security level permitted by the agent with Set runtime security level selections. If security alerts pop up as a user runs an agent manually on behalf of someone who should have complete rights for the particular actions, check the user's ECL settings.

Obviously, this means that access to the LS Admin Process database, or one designed for similar purposes, should be carefully controlled so that only authorized persons have access.



[ ] Run as web user

Run on behalf of: Hannah Green/Acme

[ ] Allow remote debugging

[ ] Allow user activation

Set runtime security level: (1 = most secure)

3. Allow restricted operations with full administration rigl

Default access for viewing and running this agent

[✓] All readers and above

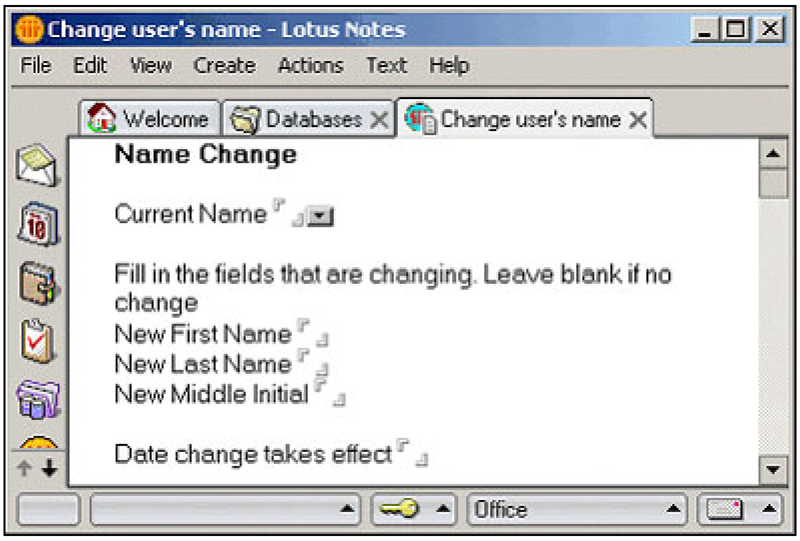
LocalDomainAdmins
LocalDomainServers
SalesReps

[ ] Allow Public Access users to view and run this agent

Another option is to schedule the agents to run on the server. (To read more about agents, see the IBM Redbook, "**Domino Designer 6: A Developer's Handbook**," and two *LDD Today* articles by Julie Kadashevich, "**Troubleshooting agents in Notes/Domino 5 and 6**" and "**Decoding the new Notes/Domino 6 agent features**.")

Renaming a user

To rename a user, start with the Name Change form in the LS Admin Process database (shown below). In the Current Name keyword field, the user—our HR rep—can pick the employee name from the Domino Directory. The HR rep then fills in the changes to any or all of the following fields, the first name, last name, or middle initial, and adds the date the change should occur. As long as the user has rights to this database, she can complete and save a document. Then, if the agent has the correct permissions and signature, as noted above, it will run on schedule.


Change user's name - Lotus Notes
File Edit View Create Actions Text Help
Welcome
Databases
Change user's name
Office

Name Change

Current Name

Fill in the fields that are changing. Leave blank if no change
New First Name
New Last Name
New Middle Initial

Date change takes effect

The following code from the 02. Rename Users agent creates collections of documents by searching for those whose date field equals the current day, whose form is called Main, and that do not yet have a field called processed. It then goes through the collection and finds the name fields that are not empty. If a field is empty, it adds an asterisk to indicate that no change should occur in that part of the name.

```
Sub Initialize
          Dim session As New NotesSession
          Dim adminp As NotesAdministrationProcess
          Dim db As NotesDatabase
          Dim doc As NotesDocument
          Dim collect As NotesDocumentCollection
          searchFormula$ = |Date = @Today & Form = "Main" & !@isAvailable(Processed)|
          Set db = session.CurrentDatabase
          'Makes a collection of docs that are dated to change name at today's date
          Set collect = db.Search(searchFormula$,Nothing,0)
          Set doc = collect.GetFirstDocument
          Do While Not (doc Is Nothing)
```

```
                        ThisName$ = doc.Name(0)
                        NewFirst$ = doc.NewFirstName(0)
                        If NewFirst$ = "" Then
                                '* indicates no change
                                NewFirst$ = "*"
                        End If
                        NewLast$ = doc.NewLastName(0)
                        If NewLast$ = "" Then
                                NewLast$ = "*"
                        End If
                        NewMI$ = doc.NewMI(0)
                        If NewMI$ = "" Then
                                NewMI$ = "*"
                        End If
```

The agent then continues by creating an instance of the NotesAdministrationProcess class. In this example, we are not using a CA, but instead use the certifier file. Note that this requires entering the path to the cert.id and the password into the agent where someone can read it. When we use the CA, it is not necessary to indicate the location of the cert.id or its password.

```
                        Set adminp = session.CreateAdministrationProcess("Pro/DeJean")
                        Adminp.CertifierFile = "C:¥notes6¥data¥cert.id"
                        adminp.CertifierPassword = "passwordhere"
```

The agent then uses the RenameNoteUser method to process the name change. NoteID$ refers to the document created in the Administration Request database by this request.

```
                        noteID$ = adminp.RenameNotesUser(ThisName$,NewLast$,NewFirst$,NewMI$)
'following lines not required but opens the doc created in the Admin Requests database.
                                If noteID$<> "" Then
                                        Dim dbadmin As New NotesDatabase("Pro/DeJean", "admin4.nsf")
                                                        Dim ws As New NotesUIWorkspace
                                        Call ws.EditDocument(False, dbadmin.GetDocumentByID(noteID$))
                                End If
```

The code above opens the Administration Request database to the document just created by the agent. This is optional—include it if you want to see the document. In most cases, you will probably omit it.

```
                        Set doc= collect.GetNextDocument(doc)
                Loop
                Call collect.StampAll("Processed","Yes")
End Sub
```

The final part of the code loops to the next document in the collection and when completed, stamps the collection as processed, so the agent does not process the documents again. The same agent could be run using a CA. In that case, the following code:

```
                    Adminp.CertifierFile = "C:¥notes6¥data¥cert.id"
                            adminp.CertifierPassword = "passwordhere"
```

is replaced with the following (for a complete example, see the agent named 02A. Rename a user with CA in the example database):

```
            If adminp.IsCertificateAuthorityAvailable = True Then
                    adminp.CertificateAuthorityOrg = "/DeJean"
                            adminp.UseCertificateAuthority = True
            End If
```

The property IsCertificateAuthorityAvailable in the first line checks to see if the Certificate Authority (CA) is available and returns a Boolean True if it is. However, it does not verify that the CA task is set up and running on the server. If it is not running, the agent cannot rename the user. If False, a certifier ID file is needed. If True, it then uses the CertificateAuthorityOrg to name the particular certifier in the CA and finally UseCertificateAuthority is set to True. Using the CA protects the certifiers, which reside on the server, from being directly accessed.

Signing a database

Many organizations sign databases ready for production with a specific ID. That ID can be the server's ID or it can be another ID. Some organizations create special IDs just for signing production databases. The Administrator often performs this task. It used to be done one database at a time in the Administrator client. The following script adds the sign database request to the Administration Requests database. (At this time, the SignDatabaseWithServerID method only works with the server ID. You cannot specify another ID to sign the database—something that would be a valuable enhancement in a future release.)

The form Sign Database allows the user to enter the path and name of the database to be signed and the server on which it resides. The script is called 03. Sign a Database.

Note that again the variable AdminP is created as a NotesAdministrationProcess and instantiated. The method this time is SignDatabaseWithServerID(server$,dbfile$[updateonly]).

```
Sub Initialize
            Dim session As New NotesSession
            Dim adminp As NotesAdministrationProcess
            Dim thisdb As NotesDatabase
            Dim view As NotesView
            Dim doc As NotesDocument

            Set thisdb =session.CurrentDatabase
            Set view = thisdb.GetView("SDB")
            Set doc = view.GetFirstDocument
```

```
        ThisDbPath$= doc.DBFilePath(0)
        ThisServer$= doc.Server(0)
        Set adminp = session.CreateAdministrationProcess(ThisServer$)
        noteID$ = adminp.SignDatabaseWithServerID(ThisServer$,ThisDbPath$,True)
End Sub
```

The updateonly parameter is set to True, so the ID signs only design elements that have changed since the last time the ID signed the database. Setting it to False signs all the design elements.

Deleting a user

Writing a script using the NotesAdministrationProcess class allows an administrator to delegate authority to terminate an employee without giving that person direct access to the Domino Directory or the Administration Requests database—two databases in which most users should not be able to edit documents.

In this example, we add a request to delete a user using the Termination form. In this form, the user selects the name of the person to be terminated, looked up from the Directory, adds the termination date, and checks whether or not the employee's mail file should be deleted along with his or her ID.

The 04. Delete User and Remove Mail agent then gets the values of fields in the form and uses the DeleteUser method to put in a request to delete the person. In the line Call adminp.DeleteUser(ThisUser$, False, DM%,"", False), ThisUser$ is the user name entered in the Termination form. False indicates that removing the user does not have to occur immediately. DM% has three possible values, indicating the disposition of the user's mail file:

- 0, or the mnemonic MAILFILE_DELETE_NONE, indicates that the user's mail file is not to be deleted
- 1, MAILFILE_DELETE_HOME, indicates the user's mail file on his or her home server is to be deleted
- 2, MAILFILE_DELETE_ALL, indicates that the mail file should be deleted from all mail replicas

The "" indicates that there is no deny list group to which the user should be added, and the final parameter is False because we are not deleting the corresponding user from Windows.

```
        Dim session As New NotesSession
        Dim adminp As NotesAdministrationProcess

        Dim thisdb As NotesDatabase
        Dim view As NotesView
        Dim doc As NotesDocument

        Set thisdb =session.CurrentDatabase
        Set view = thisdb.GetView("TE")
        Set doc = view.GetFirstDocument
        ThisUser$ = doc.Name(0)
        DeleteMail = doc.MailDel(0)
        If DeleteMail = "No" Then
            DM% = 0
        Else
            DM% = 2
        End If
        Set adminp = session.CreateAdministrationProcess("pro/DeJean")
        Call adminp.DeleteUser(ThisUser$, False, DM%,"", False)
        If noteID$<> "" Then
            Dim db As New NotesDatabase("Pro/DeJean", "admin4.nsf")
            noteID$ = adminp.ApproveDeletePersonInDirectory(noteID$)
        End If
End Sub
```

Note that the agent also uses the ApproveDeletePersonInDirectory method. Deleting a person's mail file may require an approval before it can be carried out. In addition, when this request is carried out, it removes the user from ACLs and Readers and Authors fields (depending on the ACL settings) as well as from the Domino Directory.

You can use either the numerical values for DM%, and the code above does, or the mnemonics:

```
                If DeleteMail = "No" Then
                    DM% = MAILFILE_DELETE_NONE
                Else
                    DM% = MAILFILE_DELETE_ALL
                End If
```

Recertifying a user

Recertifying a user (or a server) is another task that requires access to the organization certifier or the CA. The following agent code checks for the existence of the Certificate Authority. If it exists and is available, the agent uses the CA. If the CA does not exist or if the Certificate Authority Organization is not available, the agent uses a certifier located in a file. In this code sample (named 05. Recertify with Certifier file or CA) it is necessary to supply the certifier location and password, which can be a security risk.

```
            searchFormula$ = |Date = @Today & Form = "RCU"|
            Set db = session.CurrentDatabase
            'Makes a collection of docs that are scheduled to be processed on today's date
            Set collect = db.Search(searchFormula$,Nothing,0)
            Set doc = collect.GetFirstDocument
            Do While Not (doc Is Nothing)
                ThisName$ = doc.Name(0)
                ThisCert$ = doc.OUC(0)
                Set adminp = session.CreateAdministrationProcess("Pro/DeJean")
                'Checks if CA exists
                'names specific ceritficate in CA
```

```
                    adminp.CertificateAuthorityOrg = ThisCert$
                    If adminp.IsCertificateAuthorityAvailable = True Then
                        adminp.UseCertificateAuthority = True
                        noteID$ = adminp.ReCertifyUser(ThisName$)
                    Else
                        Call USEFILE
                    End If
                        Set doc= collect.GetNextDocument(doc)
                        Loop
                    Call collect.StampAll("Processed","Yes")
            End Sub
            Sub USEFILE
                Adminp.CertifierFile = "C:¥notes6¥data¥ids¥certs¥wcoast.id"
                    adminp.CertifierPassword = "putpasswordhere"
                    noteID$ = adminp.ReCertifyUser(ThisName)
            End Sub
```

You can adapt this script to lookup the People/Certification view in the Domino Directory and to automatically recertify users before their expiration date.

Adding members to a group

The AddGroupMembers(group$,members) method allows you to add one or more members to a group in the Domino Directory. If the group does not exist, it creates it as a multi-purpose group with the members included. If a person or group is already listed within the group, it is not duplicated. Note that if there is only one person in the EmpNames field you must create a two-element array and then call the element at index 0. The main part of the sample agent named 06. Add Group Members is shown below.

```
Set thisdb =session.CurrentDatabase
        Set view = thisdb.GetView("GADD")
        Set adminp = session.CreateAdministrationProcess("Pro/DeJean")
        Set doc = view.GetFirstDocument
        Do While Not (doc Is Nothing)
            'Get value of Group name
            ThisGroup$ = doc.GroupName(0)
            If ThisGroup$ = "" Then
                Messagebox "Missing group. You must enter the name of a group."
                    Exit Sub
            End If
                'Get values in EmpNames field
            If Ubound(doc.EmpNames) >0 Then
                noteid$ = adminp.AddGroupMembers(ThisGroup$, doc.EmpNames)
            Else
```

```
                Dim Employees(1) As String
                Employees(0) = doc.EmpNames(0)
                noteid$ = adminp.AddGroupMembers(ThisGroup$, Employees)
        End If
                Set doc = view.GetNextDocument(doc)
Loop
```

Renaming a group

In the Domino Directory, a group can also be renamed. In the following example, the Sales Representatives group is renamed to Sales Persons. (This is included in the examples database as 08. Renaming a Group.)

```
Sub Initialize
        Dim session As New NotesSession
        Dim adminp As NotesAdministrationProcess
        Set adminp = session.CreateAdministrationProcess("Pro/DeJean")
        noteID$ = adminp.RenameGroup("Sales Representatives","Sales Persons")
End Sub
```

Finding groups and users

The FindGroupInDomain(group$) and FindUserInDomain(user$) methods locate groups or users in the Domino Directory, in ACLs (which have the Administration server selected on the advanced tab), and in Policy documents. The results appear in the Administration Requests database with doclinks to each occurrence. The agent 09. Find a User is handy when you want to find all instances of a user who has been entered in an ACL as a person, but who should be in a group.

```
Sub Initialize
        Dim session As New NotesSession
        Dim adminp As NotesAdministrationProcess
        Dim thisdb As NotesDatabase
        Dim view As NotesView
        Dim doc As NotesDocument
        Set thisdb =session.CurrentDatabase
        Set view = thisdb.GetView("SDB")
        Set doc = view.GetFirstDocument
        ThisServer$ = thisdb.Server
        Set adminp = session.CreateAdministrationProcess("Pro/DeJean")
        noteID$ = adminp.FindUserInDomain("Holly Green")
        If noteID$<> "" Then
                Dim db As New NotesDatabase(ThisServer$, "admin4.nsf")
                Dim ws As New NotesUIWorkspace
                Call ws.EditDocument(False, db.GetDocumentByID(noteID$))
        End If
End Sub
```

Changing an HTTP password

This simple script changes the HTTP password for Robert Smith from password to newpassword123. Its usefulness is limited. If you don't know the original HTTP password, you cannot change it to a new one. (This agent is included in the examples database as 10. Change an HTTP password.)

```
Sub Initialize
    Dim session As New NotesSession
    Dim adminp As NotesAdministrationProcess
    Set adminp = session.CreateAdministrationProcess("Pro/DeJean")
    noteID$ = adminp.ChangeHTTPPassword("Robert Smith", "password",
    "newpassword123")
```

Changing a user's password settings

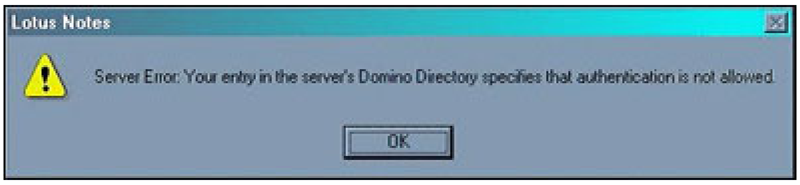
The following code samples, using the SetUserPasswordSettings method, are useful when you want to lockout a user or force a password change.

Note: For this method to work, the Server document must have password checking enabled. The script makes an Administration Request to change three fields in the Person document in the Domino Directory. The fields are Check password, Required change interval, and Grace period.

This script (included in the examples database as 11. Change user password settings) is useful if an employee is out on disability, and you want to temporarily disable their access, or to deny contractors hired for a fixed period access to servers after their contracts expire. It allows for forcing password changes at particular intervals.

The method and parameters are SetUserPasswordSettings(username$ [, notespasswordchecksetting$] [, notespasswordchangeinterval$] [, notespasswordgraceperiod$] [, internetpasswordforcechange$])

In this example, the user, Peter Purple, is unable to access databases on the server after the agent has run. When he tries, he receives the following error message.


Lotus Notes

Server Error: Your entry in the server's Domino Directory specifies that authentication is not allowed.

The agent changes three fields in the Person document in the Domino Directory.

The notespasswordchecksettings$ parameter has three possible choices that act on the Check password field:

- PWD_CHK_CHECKPASSWORD sets the Check password field to Check password.
- PWD_CHK_DONTCHECKPASSWORD sets the Check password field to Don't check password.
- PWD_CHK_LOCKOUT sets the Check password field to Lockout ID.

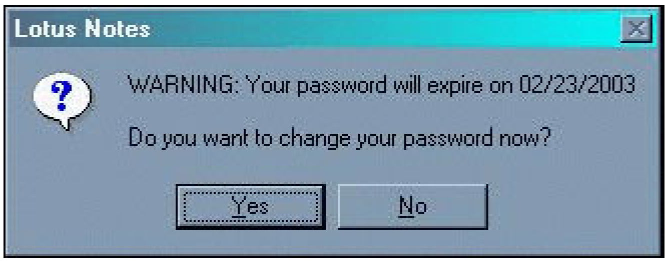
The notespasswordchangeinterval changes the Required change interval field. Zero indicates no interval has been set. The notespasswordgraceperiod changes the Grace period field. Zero indicates no grace period. The internetpasswordforcechange parameter is Boolean. True means the user is asked to change his password the next time he logs in; False means is not asked.

```
Dim session As New NotesSession
Dim adminp As NotesAdministrationProcess
Set adminp = session.CreateAdministrationProcess("Pro/DeJean")
noteID$ = adminp.SetUserPasswordSettings("CN=Peter Purple/O=DeJean",PWD_CHK_LOCKOUT,
0, 1, True)
```

Alternatively, if the parameters are changed to the line below, the user is forced to change his password when he next logs in.

```
noteID$ = adminp.SetUserPasswordSettings("CN=Peter
Purple/O=DeJean",PWD_CHK_CHECKPASSWORD, 1, 1, True)
```

He receives a prompt like the one below.


Lotus Notes

WARNING: Your password will expire on 02/23/2003

Do you want to change your password now?

Conclusion

The NotesAdministrationProcess contains other methods not covered in this article. You can use many of these methods, including AddInternetCertificateToUser, MoveMailUser, MoveUserInHierachyRequest, and UpgradeUserToHierarchical, in scripts that will be of benefit to your organization's administrators.

To use the NotesAdministrationProcess classes effectively and efficiently, you first need to understand how your organization functions and where and to whom you can distribute tasks. The Domino developers and administrators should get together to determine where scripts using the NotesAdministrationProcess class can be most useful in their organizations. The new class offers developers and users an opportunity to have authority over their own areas of expertise without endangering the security that the Domino environment provides. It provides relief to administrators, allowing them to delegate responsibility for some tasks to others without granting them unrestricted access to the administrator client.

ABOUT THE AUTHORS

Sally Blanning DeJean and David DeJean have been working with and writing about Lotus Notes and Domino for as long as they've existed. They were co-authors of the very first book about Notes, *Lotus Notes at Work.* Sally, a CLP

Principal, has written other books about Notes and is a full-time Notes/Domino developer. David, a CLP, has been an editor and writer for several computer publications. He is a partner in DeJean & Clemens, a firm that develops Notes and Internet applications and technical and marketing communications.