Users cannot send e-mail messages from a mobile device or from a shared mailbox in Exchange 2000 Server and in Exchange Server 2003

Article ID : 912918
Last Review : August 1, 2006
Revision : 10.1

On This Page

- SYMPTOMS
  - Error messages
  - NDRs
- CAUSE
- RESOLUTION
  - How to grant the Send As permission for multiple accounts
  - Special rules for adminSDHolder Protected Accounts
- MORE INFORMATION
  - Mailbox owners
  - Associated external accounts
  - Delegation scenarios
  - Finding accounts that have the Full Mailbox Access permission without the Send As permission
    - Permissions that are required for the script
    - Administrative workstation configuration for the script
    - Editing the export file
  - About mailbox owners with delegates
  - How to open an Export file in Excel
  - How to save an Export file after you edit the file in Excel
  - Script syntax
    - To obtain command line help, enter the following command:
    - To export users who have Full Mailbox Access without the Send As permission for a domain, enter the following command:
    - To import an edited Export file, enter the following command:
  - How to grant the Send As permission respectively for each mailbox in the domain for all users who already have the Full Mailbox Access permission for a mailbox
    - Script modifications
  - Tips and caveats

SYMPTOMS

When you try to send an e-mail message in Microsoft Exchange 2000 Server or in Microsoft Exchange Server 2003, you cannot send the e-mail message. Additionally, you may receive one of the following error messages or one of the following Non Delivery Reports (NDRs):

Error messages

- **Access denied**
- **You do not have sufficient permission to perform this operation on this object. See the folder contact or your system administrator.**
- **Unlisted Message Error**
- **MAPI_E_NO_ACCESS -2147024891**
- **Failed to submit mail message for user USERNAME (HRESULT:-2147024891) Pausing user USERNAME. (Security error - Cannot access the users mailbox.)**

- **Resource Not Found**

Note You receive the "Access denied" error message or the "Resource not found" error message from Outlook Web Access (OWA) when you are logged in as a delegate account.

NDRs

- You do not have permission to send to this recipient. For assistance, contact your system administrator.
- The message could not be sent using your mailbox. You do not have the permission to send the message on behalf of the specified user.

This issue is known to affect the following third-party products:

- Research In Motion (RIM) Blackberry Enterprise Server (BES)
- Good Technology GoodLink Wireless Messaging

However, it has been confirmed that the following third-party products are not affected by this issue.

- Cisco Unity Unified Messaging
- Quest Migration Suite for Exchange
- The Microsoft ExMerge utility for Exchange

This issue may also affect custom MAPI or Collaborative Data Objects (CDO)-based programs that send e-mail messages.

Other third-party products that use service accounts to send e-mail messages may also be affected. If you are running a third-party product that is affected by this issue, we recommend that you contact the vendor for help with resolving this issue. For more information, see the "More Information" section.

Important As an emergency measure to restore Send As functionality for business critical applications, you can grant Send As permission to a service account through inheritance on an Active Directory container or even an entire domain. To do this, see the detailed instructions in the section. Although this is an effective way to resolve the immediate problem, there are security and management implications that you should consider. This grant may apply the Send As permission for accounts that you did not intend, and you must consider the Send As permission if you move user accounts to a different container.

CAUSE

This issue may occur when one of the following conditions is true:

- You do not have permissions to send e-mail messages as the mailbox owner in the account that you are using to send the e-mail message.
- You are running Microsoft Exchange 2000 Server Service Pack 3 (SP3) with a Store.exe file version that is equal to or later than version 6619.4. Version 6619.4 was first made available in the following Microsoft Knowledge Base article:

  915358 (http://support.microsoft.com/kb/915358/) A hotfix is available to change the behavior of the Full Mailbox Access permission in Exchange 2000 Server

- You are running Microsoft Exchange Server 2003 Service Pack 1 (SP1) with a Store.exe file version that is equal to or later than version 7233.51. Version 7233.51 was first made available in the following Microsoft Knowledge Base article:

  895949 (http://support.microsoft.com/kb/895949/) "Send As" permission behavior change in Exchange 2003

  Note that this fix is not included with Microsoft Exchange 2003 Service Pack 2 (SP2). If you have installed the Exchange Server 2003 SP1 version of this hotfix, you must install the Service Pack 2 version after you upgrade to Service Pack 2.

- You are running Exchange Server 2003 SP2 with a Store.exe file version that is equal to or later than version 7650.23. Version 7650.23 was first made available in the following Microsoft Knowledge Base article:

895949 (http://support.microsoft.com/kb/895949/) "Send As" permission behavior change in Exchange 2003

Note This change was not included in Exchange 2000 Server SP3, in Exchange Server 2003 SP1 or in Exchange 2003 SP2. The change was implemented after release of all of these service packs, but it is supported in each of them. The change will be included in future service packs for these products.

If you install Exchange Server 2003 SP2, you must install the additional update to retain the new behavior, even if you already installed the version of the update for Exchange Server 2003 SP1.

RESOLUTION

Before the Store.exe file versions that are listed in the "Cause" section, granting the Full Mailbox Access permission implicitly granted permission to send as the mailbox owner. This meant that another account that has the Full Mailbox Access permission could send e-mail messages that appeared as if they were sent by the mailbox owner.

Many Microsoft Exchange customers have requested that Send As permission be separated from the Full Mailbox Access permission for the following two reasons:

- To deter e-mail spoofing.
- To make sure that e-mail messages that are sent by a delegate can always be clearly distinguished from e-mail messages that are sent by the actual mailbox owner.

All new versions of the Exchange Information Store will now explicitly require the Send As permission in order to send e-mail messages as the mailbox owner. However, the following lists the three exceptions to this requirement:

- The mailbox owner account does not require explicit Send As permission for its own mailbox.
- The Associated External Account for a mailbox does not require explicit Send As permission.
- A delegate account that also has the Full Mailbox Access permission does not require explicit Send As permission.

For more detail about these exceptions, see the "More Information" section.

All other accounts that are granted partial or full access to a mailbox must now be explicitly granted the Send As permission for the mailbox owner account in order to send mail as the mailbox owner. This includes application service accounts that perform functions such as sending e-mail messages for mobile device users.

The Send As permission must be granted to the service account on each user object that owns a mailbox. You cannot grant the Send As permission on an Exchange server or on a database object and achieve the effect of granting the Send As permission for all mailboxes in the database.

This behavior occurs because the Send As permission is an Active Directory permission that applies to the Active Directory objects for which it is set. Granting the Send As permission on an Exchange database object gives you permission to the Send As permission the database itself. However, it does not give you permission to the users with Send As permissions who have mailboxes in the database.

Note Granting the Receive As permission on an Exchange database is the functional equivalent of granting the Full Mailbox Access permission to all mailboxes that are in the database. This differs from the behavior of the Send As permission.

In the Send As permission, the permission applies only to the database object itself. It does not to the mailboxes in the database. In the Receive As permission, the permission is apparently inherited by all mailboxes that are in the database.

To better understand the difference between the two permissions, think of all the mailboxes in a database as if they were folders in a single mailbox (the "database" mailbox). If you have full access to the database, you have permission to access all the contents in the database. This includes all the mailboxes.

The Send As permission, applies to the identity of an Active Directory user object, not to mailbox contents stored in a database. When e-mail messages are sent, they are not sent from a particular mailbox or database, but from a user. The user may be the mailbox owner or any other account that has the Send As permission.

To explicitly grant another account permission to send as a mailbox owner, follow these steps:

1. Start the Active Directory Users and Computers management console.

2. On the **View** menu, make sure that the **Advanced Features** option is selected. If this option is not selected, the Security page will not be visible for User account objects.
3. Open the properties of the user account that owns the mailbox.
4. Click the **Security** tab.
5. If the account is not already in the list of group or user names, add the account that should have the Send As permission for this user.
6. In the **Permissions** box, click **Allow for the "Send As"** permission for the appropriate account.
7. Click **OK**.

Note that it may take fifteen minutes for the Exchange Information Store to update its permissions cache and make the new permission effective.

How to grant the Send As permission for multiple accounts

A sample script is provided at the end of this article which will search an Active Directory directory service domain for accounts that have the Full Mailbox Access permission without the Send As permission for a mailbox. These are the characteristics of a service or resource account that will be affected by this security change. The script can generate an export file which you can review, edit, and then re-import to grant the Send As permission to accounts that require this permission.

You can also grant the Send As permission by inheritance on every user object in an Active Directory domain or in a container. If you grant the Send As permission by this method, you may grant permission for objects that you did not intend. Additionally, you may lose permission for objects that are moved from the container. Therefore, this method is not preferred and may have security implications which should be carefully considered before you implement it.

To grant Send As for a single account on all user accounts in an Active Directory domain or container, follow these steps:

1. Start the Active Directory Users and Computers management console.
2. On the **View** menu, make sure that the **Advanced Features** option is selected. If this option is not selected, the Security page will not be visible for domain and container objects.
3. Open the properties of the domain or container, and then click the Security page.
4. Click the **Advanced** button.
5. If the account that needs permission is not already listed, click **Add**, and then select the account. Otherwise double-click the account for editing.
6. In the **Applies Onto** list, click **User Objects**.
7. Grant the account Send As permission.
8. Click **OK** until you have exited and saved all changes.

Note The script that is described at the end of this article considers inherited permissions. Therefore, if you grant Send As permission by using this method, the accounts with inherited Send As permission will become invisible to the script. To process these accounts with the script later, you must remove the inherited Send As permission first.

Special rules for adminSDHolder Protected Accounts

If you use the script to grant the Send As permission for a mailbox owner that is also a domain administrator, the Send As permission will not be effective. We strongly recommend that you do not mailbox-enable user accounts that have domain administrator rights or that are adminSDHolder protected.

The **adminSDHolder** object is a template for accounts that have broad Active Directory administrative rights. To prevent unintended elevation of privilege, any account that is protected by the **adminSDHolder** object must have access rights that match those that are listed on the **adminSDHolder** object itself.

If you change the rights or the permissions on the **adminSDHolder** object for a protected account, a background task will undo the change within several minutes. For example, if you grant the Send As permission on a domain administrator object for an application service account, the background task will automatically revoke the permission.

Therefore, you cannot grant the Send As permission to an application service account for an account that is protected by the **adminSDHolder** object unless you change the **adminSDHolder** object itself. If you do change the **adminSDHolder** object, this will change the access permissions for all protected accounts. You should only change the **adminSDHolder** object after a complete review of the security implications that may occur with the change.

To associate a mailbox with an account that is protected by the **adminSDHolder** object, follow these steps:

1. Start the Active Directory Users and Computers management console.
2. On the **View** menu, make sure that the **Advanced Features** option is selected. If this option is not selected, the Security page will not be visible for User account objects.
3. Create an ordinary user account to act as the mailbox owner.
4. Assign the ordinary user account a mailbox on an Exchange server.
5. Open the properties of the new mailbox owner account.
6. In the **Exchange Advanced** box, grant the Full Mailbox Access permission to the protected administrator account.
7. In the Security page, grant the Send As permission to the protected administrator account.
8. Click **OK** to exit the properties of the mailbox owner object.
9. Right-click the mailbox owner account object, and then click **Disable Account** to disable the account for all logons.

For more information about adminSDHolder-protected accounts, click the following article numbers to view the articles in the Microsoft Knowledge Base:

907434 (http://support.microsoft.com/kb/907434/) The "Send As" right is removed from a user object after you configure the "Send As" right in the Active Directory Users and Computers snap-in in Exchange Server

318180 (http://support.microsoft.com/kb/318180/) AdminSDHolder thread affects transitive members of distribution groups

817433 (http://support.microsoft.com/kb/817433/) Delegated permissions are not available and inheritance is automatically disabled

306398 (http://support.microsoft.com/kb/306398/) AdminSDHolder object affects delegation of control for past administrator accounts

MORE INFORMATION

Exchange mailbox and folder access permissions are split between Active Directory and Microsoft Exchange databases. However, both kinds of permissions are set in the Active Directory user management console, but different permissions are stored in two separate locations.

Generally, if a permission is set on the Security page for an object, it is an Active Directory permission. If it is set on the Exchange Advanced Mailbox Rights page, it is an Exchange database permission. Therefore, if you try to access the Mailbox Rights page when a user’s database is unavailable, you will receive the following error message::

The Microsoft Information Store service is unavailable.

The Associated External Account permission is an exception to the rule that permissions set through the Exchange Advanced Mailbox Rights are stored in the Exchange database. The Associated External Account permission is not even a true permission, but a way of setting the Active Directory attribute **msExchMasterAccountSID**. The **msExchMasterAccountSID** attribute, while not a permission in itself, controls how other permissions work. For more detail about the **msExchMasterAccountSID** attribute, see the "Associated External Accounts" section.

Note The Active Directory attribute **msExchMailboxSecurityDescriptor** is a backup copy of a subset of the effective mailbox rights. It is used internally by Exchange for a variety of purposes. Additionally, the **msExchMailboxSecurityDescriptor** attribute is updated to match current effective rights if administrators use supported interfaces to assign rights.

However, if the **msExchMailboxSecurityDescriptor** attribute is modified directly by an administrator, the changes will not be propagated to the Exchange store and the changes will not take effect. It is not guaranteed to be synchronized with actual mailbox rights. You should not use the **msExchMailboxSecurityDescriptor** attribute to read or write mailbox rights.

Mailbox rights are written not only into an Exchange database, but also into the Active Directory attribute **msExchMailboxSecurityDescriptor**. The **msExchMailboxSecurityDescriptor** attribute is a “backup copy” of several of the effective mailbox rights. Additionally, the **msExchMailboxSecurityDescriptor** attribute is used internally by Exchange for a variety of purposes. If an administrator uses supported interfaces to assign mailbox rights, the **msExchMailboxSecurityDescriptor** attribute will be updated as rights are changed. However, if the **msExchMailboxSecurityDescriptor** attribute is modified directly by an administrator, the changes will not be propagated to the Exchange store and the changes will not take effect. Therefore, the **msExchMailboxSecurityDescriptor** attribute is not guaranteed to be synchronized with actual mailbox rights. You should not use the **msExchMailboxSecurityDescriptor** attribute to read or to write mailbox rights.

For more information, click the following article number to view the article in the Microsoft Knowledge Base:

> 310866 (http://support.microsoft.com/kb/310866/) How to set Exchange Server 2003 and Exchange 2000 Server mailbox rights on a mailbox that exists in the information store

The Full Mailbox Access permission is an Exchange database store permission. The Send As permission is an Active Directory permission. Before the Exchange Store.exe file changes that are described in this article, the Exchange system did not consult the setting for the Send As permission if the sender already had the Full Mailbox Access permission.

Note You can grant the Send As permission without granting the Full Mailbox Access permission. In these situations, Exchange has always consulted the Send As permission.

Inclusion of the Send As permission with the Full Mailbox Access permission has allowed Exchange server administrators to grant themselves effective Send As permissions for any mailbox on a server that they administer. Administrators can perform this action because the administrators have full effective control over an Exchange database. By separating the Send As permission from the Full Mailbox Access permission, Active Directory administrators can now block this process because the Send As permission is an Active Directory permission and not an Exchange store permission. Therefore, the process is not necessarily under the control of Exchange administrators.

Mailbox owners

A mailbox owner is defined as the Active Directory user account whose **msExchMailboxGUID** attribute carries the Globally Unique numeric Identifier (GUID) for a particular mailbox. Only one account in a whole forest is permitted to carry the GUID for a particular mailbox. If you try to set a second owner with the same GUID, Active Directory will reject the change with an error.

When you mailbox-enable an account or when you connect a disconnected mailbox to an Active Directory account, the mailbox GUID is automatically set on that account. It is rarely necessary or recommended for administrators to set mailbox GUIDs directly.

Associated external accounts

A common Exchange configuration is to install Exchange in a resource forest. A resource forest is a forest that is in a different forest from the user accounts that will have mailboxes in the system. This presents a problem because the **msExchMailboxGUID** attribute can only be set on objects in the same forest as the Exchange server.

The solution to this problem is to mailbox-enable an account in the Exchange server forest. Then, you link this mailbox-enabled account to one in another forest or in a Microsoft Windows NT 4 domain. You can do this by granting the Associated External Account permission. Only a single account can be granted the Associated External Account permission. The account selected must be from a different forest.

When you set the Associated External Account permission, you are writing the SID value for the external account to the **msExchMasterAccountSID** attribute of the mailbox owner. Therefore, this is not a permission at all, but a convenient way of controlling the value of the **msExchMasterAccountSID** attribute. After the **msExchMasterAccountSID** attribute has been set, the external account that owns the SID will be granted Exchange access as if it were the actual mailbox owner account.

Note that this applies only to Exchange access, not to all Active Directory access. Additionally, you should mark the mailbox owner account Disabled for logons after you set the Associated External Account permission so that all permissions work as expected.

> 300456 (http://support.microsoft.com/kb/300456/) Client permissions and delegations do not persist after being assigned in Exchange 2000

Delegation scenarios

A delegate is a user who has been granted partial access to another mailbox and the right to send e-mail messages on behalf of that mailbox owner. A common delegation scenario is to grant delegate access to an administrative assistant for an executive's calendar. The delegate can typically read and update the calendar. Additionally, the delegate can reply to any e-mail messages on behalf of the executive.

Delegate access is granted by adding the delegate to the multi-valued **publicDelegates** attribute of the mailbox owner. All users listed in this attribute have the Send on Behalf Of permission for the mailbox owner. When such delegates send an e-mail message that has the owner’s name in the **From** box, the e-mail message’s **From** box displays the following value:

<*Delegate’s Name*> on behalf of <*Mailbox Owner*>

However, the e-mail message is sent from the delegate and not from or as the mailbox owner.

The following is a list of the two interfaces that you can use to grant the Send on Behalf Of and delegate permissions:

- On the mailbox owner object, grant the Send on Behalf permission in the **Exchange General** dialog box.
- In Microsoft Outlook, use the **Delegates** dialog box.

Both of these methods set the **publicDelegates** attribute. However, the Outlook method also allows for granting specific folder permissions to the delegate. You can also grant permissions to a delegate directly from the properties of an individual folder in Outlook.

In some cases, you may be unable to set the **publicDelegates** attribute in Outlook.

329622 (http://support.microsoft.com/kb/329622/) "Send on behalf" permission is not assigned to a user after you delegate access in Outlook

If you grant delegate access to your mailbox, the delegate can use the Send On Behalf permission even if you do not grant access to any one of your mailbox folders. The fundamental permission a delegate has is the Send on Behalf Of permission. Permissions to access your mailbox folders are separate and must be granted in addition to delegation permissions.

Typically, delegates will use Microsoft Outlook to access individual folders to which you have given them permission. You can do this by clicking **Open** on Outlook **File** menu, and then clicking **Other User’s Folder**.

Alternately, delegates can open your mailbox by listing it as an additional mailbox in the **Advanced** tab of their Outlook profiles. This method causes your mailbox to be listed in the delegate's Outlook folder tree. Additionally, this method allows for access to all folders in your mailbox for which a delegate has been granted permissions.

You may want your delegate to sometimes have the Send on Behalf of permission and at other times have the Send As permission. To configure a delegate with these two permissions, follow these steps:

- Grant the delegate the Full Mailbox Access permission. This cannot be done through Outlook. Instead, an Active Directory administrator must do this on the mailbox owner account. Even if you grant Owner permissions on every folder in your mailbox, that is not the same permission as the Full Mailbox Access permission.
- Do not grant the delegate the Send As permission. If you grant the delegate the Send As permission, all e-mail messages that are sent by the delegate will be done with the Send As permission. The delegate will no longer be able to use the Send on Behalf Of permission.

In this scenario, delegates that want to use the Send on Behalf Of permission should log on to their own mailboxes. If the delegates are replying to or forwarding an e-mail message that is already in one of your folders, the e-mail message will automatically be sent on behalf of you. If the delegates create a new e-mail message on your behalf, they must enter your name in the **From** box for the e-mail message to be sent on your behalf.

Regardless of whether the delegates have opened your folders or your whole mailbox as a secondary mailbox, all e-mail messages that they send from you will use the Send on Behalf Of permission as long as their own mailbox is the primary mailbox for the current Outlook profile.

When delegates want to send an e-mail message as you, they should log on to your mailbox with a separate Outlook profile that opens only your mailbox. E-mail messages that the delegates send while they are logged on to this profile will automatically be sent from you.

Finding accounts that have the Full Mailbox Access permission without the Send As permission

The sample script that is described in this section can search one Active Directory domain at a time for user accounts where the Full Mailbox Access permission has been granted to a mailbox without the Send As permission.

Important Before you change permissions, see the About mailbox owners with delegates section.

The script has the following three modes:

- **Export:** You can output a list of users who have the Full Mailbox Access permission but not the Send As permission. You can then review this list in Notepad or another editor to remove any accounts that you do not want to have the Send As permission.
- **Import:** You can import a list of users who have the Full Mailbox Access permission to whom the Send As permission should also be granted. Note that you cannot use this script to grant both the Full Mailbox Access permission and the Send As permission. Each account must already have the Full Mailbox Access permission in order to be granted the Send As permission.
- **SetAll:** You can grant the Send As permission to all users in the domain who already have the Full Mailbox Access permission for a particular mailbox. A log file will be generated in the same format as the Export file. This is equivalent to running the Export and the Import modes without editing the Export file.

Note There is no undo function in this script.

Permissions that are required for the script

You must run the script while you are logged on with an administrative account that is from the same forest that the mailbox owner accounts are from. The script may not work with an account that has cross-forest administrative permissions. The script may also not work when you run it from a workstation that is joined to a different forest than the forest to which the mailbox owner accounts are joined.

Given these conditions, you can run the script with multiple administrative accounts in a single logon session by using the RunAs.exe command. This procedure may be useful if you have segmented Active Directory and Exchange Server permissions and you have no single account that can administer all Exchange servers or all Active Directory domains. You can open a command prompt to run the script as each administrative account. Consider the following example:

```
RunAs.exe /user:domain\account CMD.EXE
```

Note You should not run multiple copies of the script at the same time against the same domain.

The fields that are in the Export file are as follows. The fields are described in the order that they are presented in the Export file.

- **Display name of the mailbox owner account**

  There may be more than one line in the output file that lists the same mailbox owner. This behavior occurs when multiple other accounts have Full Mailbox Access permission to the same mailbox.
- **Domain and logon name of an account that has the Full Mailbox Access permissions but not the Send As permission**

  The same account may appear multiple times throughout the Export file when the account has access to multiple mailboxes. This is likely to be the case for an application service account or for a person who manages multiple resource mailboxes.
- **Display Name of an account that has the Full Mailbox Access permission but not the Send As permission**

  This field is provided in addition to the Logon Name field to make it easier for you to identify the account.
- **Delegate Status of the mailbox owner**

  If the mailbox owner has delegates, the field value is Has Delegates. If the mailbox owner has no delegates, the field value is No Delegates.
- **Enabled or disabled status of the mailbox owner account**

  This field is useful when you want to identify resource accounts or cross-forest mailbox accounts. Typically, these accounts are disabled.
- **Full Distinguished Name of the mailbox owner account**

  This field is useful when you want to identify the domain and the container of the mailbox owner account.
- **Full Distinguished Name of the mailbox owner's mailbox database**

  This field includes the database, the storage group, the server, and the administrative group for the mailbox.

In the following example, the user who has the logon name "NoSendAs" has the Full Mailbox Access permission but not the Send As permission for the "Mailbox Owner" mailbox:

```
"""Mailbox Owner""" """Domain\NoSendAs""" """No Send As User""" """Has Delegates"""
"""Enabled""" [additional fields omitted]
```

Administrative workstation configuration for the script

This script uses Exchange management interfaces to communicate with Exchange servers. Therefore, this script must be run from an Exchange server or from a workstation with Exchange System Administrator installed.

Editing the export file

The export file is formatted as Unicode plain text so that character sets from multiple languages can be accommodated. Some text editors may not be able to correctly view or edit the file or may save the file as ANSI or ASCII text. The Notepad utility for Microsoft Windows XP, for Microsoft Windows 2000, and for Microsoft Windows 2003 can correctly handle Unicode text files. Additionally, Microsoft Excel can correctly handle Unicode text files.

The output file is in a tab-delimited format with triple quotation marks around the values for each field. The triple quotation marks are used to make importing and exporting from Excel more deterministic. In Excel, the triple quotation marks will become single quotation marks, and will revert to triple quotation marks when the file is saved again as Unicode text. See the following instructions to correctly open and save an Export file in Excel.

You can also filter an export file without using Excel by using the the Find.exe utility or the Findstr.exe utility. These utilities are included with Windows. They allow you to search for words in a file and to output only lines that contain those words or only lines that do not contain those words. For example, if you want to make a list of all mailbox owners in the file that have delegates, use either of these commands to create a file that contains only lines with the string “Has Delegates”:

Find.exe “Has Delegates” OriginalFile.txt > HasDelegates.txt

Findstr.exe /C:”Has Delegates” OriginalFile.txt > HasDelegates.txt

As another example, suppose that you filter out all the mailbox owners with delegates. The **/V** switch outputs all lines that do not match the search words. You can use any of these commands to generate a file that excludes all “Has Delegates” lines:

Find.exe “No Delegates” OriginalFile.txt > NoDelegates.txt

Find.exe /V “Has Delegates” OriginalFile.txt > NoDelegates.txt

Findstr.exe /C:”No Delegates” OriginalFile.txt > NoDelegates.txt

Findstr.exe /V /C:”Has Delegates” OriginalFile.txt > NoDelegates.txt

You can also use these commands to generate a file that lists all the accounts where an application service account has Full Mailbox Access permission but does not have Send As permission. The **/I** switch makes the command case insensitive:

Find.exe /I “domain\ServiceAccount” OriginalFile.txt > ServiceAccount.txt

Findstr.exe /I /C:”domain\ServiceAccount” OriginalFile.txt > ServiceAccount.txt

Note If you use Find.exe to generate a filtered file, you must remove the header lines that Find.exe will create at the top of the file.

Do not use wildcard filenames (*.*) with Findstr.exe. If you do use wildcards, each line in the output file will be prefaced by the file name. You should examine the output file carefully after you filter with Find.exe or with Findstr.exe to verify that your filter captured or excluded the accounts you intended.

In the following example, the user who has the logon name "NoSendAs" has the Full Mailbox Access permission but not the Send As permission for the "Mailbox Owner" mailbox.

```
"""Mailbox Owner""" """Domain\NoSendAs""" """No Send As User""" """Has
Delegates""" """Enabled""" [additional fields omitted]
```

About mailbox owners with delegates

A delegate who has Full Mailbox Access (also known as a "super-delegate") usually should not be granted Send As permission. When the super-delegate logs directly on to the mailbox owner's mailbox, the delegate can Send As.

When the delegate uses Outlook's delegation features (Additional Mailboxes to Open or Open Other User's Folder), messages are sent On Behalf Of.

Grant Send As permission to a super-delegate only if you want the delegate always to Send As the mailbox owner and never to Send on Behalf Of the mailbox owner. We recommend that you search the export file for the text "Has Delegates" and then determine whether any one of the super-delegates that are listed are in fact delegates of the mailbox owner.

Only super-delegates are listed in the export file. Ordinary delegates do not have Full Mailbox Access permission. Additionally, when you grant Send As permission to an ordinary delegate, the delegate will always Send As the mailbox owner. This is true even when the ordinary delegate does not have Full Mailbox Access. If you do grant Send As permissions to a delegate when you did not intend to, you can easily revoke the permission later.

How to open an Export file in Excel

1. Start Excel before you open the Export file.
2. Open the file in Excel as type Text Files. This will start the Text Import Wizard.
3. In the Text Import Wizard, change or accept the following settings:
   - Original data type: Delimited
   - Start import at row: 1
   - File Origin: Unicode (UTF-8)
   - Delimiters: Tab only
   - Treat consecutive delimiters as one: unmarked
   - Text qualifier: " (double quotation marks)

How to save an Export file after you edit the file in Excel

1. In the Export file, click **Save As**.
2. Give the file a different name so that you preserve an unedited copy of the original file.
3. Click **File**, click **Save As**, enter a file name for saving the output, and then select **Unicode text** in the **Save As Type** dropdown.

Script syntax

This is a text mode script and it should be run at a Command Prompt window, not from the **Run** dialog box. To open a Command Prompt window, click **Start**, click **Run**, type **CMD** in the **Open** box, and then click **OK**.

Error log and Export files will be saved to the current command prompt directory. You must have permissions to create files in this directory.

To obtain command line help, enter the following command:

```
CSCRIPT AddSendAs.vbs
```

To export users who have Full Mailbox Access without the Send As permission for a domain, enter the following command:

```
CSCRIPT AddSendAs.vbs [domain controller name] –Export
Example:
CSCRIPT AddSendAs.vbs CORP-DC-1 –Export
```

The export file will be generated as "Send_As_Export_H_MM_SS.txt".

To import an edited Export file, enter the following command:

```
CSCRIPT AddSendAs.vbs [domain controller name] –Import [filename]
```

```
Example:

CSCRIPT AddSendAs.vbs CORP-DC-1 –Import "Send_As_Export_H_MM_SS.txt"
```

How to grant the Send As permission respectively for each mailbox in the domain for all users who already have the Full Mailbox Access permission for a mailbox

Note If you have delegates who also have the Full Mailbox Access permission in your organization, you should not use the SetAll mode. If you do use the SetAll mode in this situation, delegates will be granted the Send As permission. This behavior can cause all e-mail messages that they send to use the Sent As permission instead of the Sent on Behalf Of permission. You can correct this behavior by removing the Send As permission that was mistakenly granted to the delegate.

```
CSCRIPT AddSendAs.vbs [domain controller name] –SetAll

Example:

CSCRIPT AddSendAs.vbs CORP-DC-1 –SetAll
```

If you use the SetAll mode, an export file will be generated as Send_As_Export_H_MM_SS.txt. You should save this file because it is a record of all the accounts that were changed. If you were to run the script again, it would not output the same list of accounts because the accounts would already have been granted the Send As permission.

Errors that you experience while you are running the script will be saved to the Send_As_Errors_H_MM_SS.txt file. The error file name will match the hours_minutes_seconds timestamp of any associated export file.

Script modifications

There may be accounts in your organization that have permissions on many objects, but you do not want to alter the permissions. To reduce the size of the export file, you can filter these accounts by modifying the FMA_EXCLUSIVE_LIST variable that is located near the top of the script. By default, this variable lists a few accounts that should be suppressed in the script output. You can add more accounts by using the following format:

```
& "<Domain\Name>" & OUTPUT_DELIMITER
```

For example, you could change the value of the following variable:

```
FMA_EXCLUSIVE_LIST = OUTPUT_DELIMITER & "NT AUTHORITY\SELF" & OUTPUT_DELIMITER & "NT
AUTHORITY\SYSTEM" & OUTPUT_DELIMITER
```

so that it appears as follows:

```
FMA_EXCLUSIVE_LIST = OUTPUT_DELIMITER & "NT AUTHORITY\SELF" & OUTPUT_DELIMITER & "NT
AUTHORITY\SYSTEM" & OUTPUT_DELIMITER & "Mydomain\Service1" & OUTPUT DELIMITER
```

Note These lines have been wrapped for readability. However, they should appear as a single line in the script.

This change suppresses the listing of the "Mydomain\Service1" account in the export file together with "NT AUTHORITY\SELF" and "NT AUTHORITY\SYSTEM." Notice that the Domain\Name value is case-sensitive, and it must appear exactly as it does or as it would in the export file.

There is another editable variable, FMA_EXCLUSIVE_EXSVC that has the default value "\Exchange Services" & OUTPUT_DELIMITER. "Exchange Services" is the name of an account that is granted permissions through the Active Directory Connector in Exchange 5.5 and in Exchange 2000 migration and co-existence scenarios. This account is created in multiple domains, and it may appear repeatedly in the export file if it is not suppressed.

The FMA_EXCLUSIVE_EXSVC variable accepts only one account as its value. The account name is not case-sensitive. The account that is listed must start with a backslash character (\) and should not include the domain to which the account belongs. The account will be suppressed for all domains in which it exists.

If you have used third-party migration tools or directory synchronization methods, a different account may exist in multiple domains that has widely-granted permissions to user mailboxes. In this scenario, you can substitute the name of that account for "\Exchange Services."

Tips and caveats

- Do not discard log and error files that are generated by the script. They may be valuable for troubleshooting or reversing changes later. Remember, as soon as you have granted the Send As permission to an account, it will no longer be logged in the export file.
- If an Exchange server or database is down, this will slow script processing. In such a case, you can sort the Export file by database and move lines associated with a stopped database to a different file for later import.
- The script suppresses output of accounts where the logon name ends in "$" or is NT AUTHORITY\SYSTEM. These system accounts should not typically need the Send As permission, and removing them from the Export file greatly reduces its size.
- The Export file must be in Unicode format before it can be imported. If you have unintentionally saved the file as ANSI text, you can correct this problem by loading the file in Notepad and saving it as Unicode text.
- If import is not working, troubleshoot with test accounts and a single line in the import file. You should configure a test account that has a mailbox on a running Exchange server, and then grant another test account the Full Mailbox Access permission, but not the Send As permission.
- There is no Undo mode for this script. To take away the Send As permissions you have granted with this script, you must generate another script or remove them manually. An Undo mode is not provided to avoid use of this script to remove the Send As permissions for all users in an organization.
- The script does not correctly handle an account that has been granted Full Control of a user object together with Full Mailbox Access. Full Control includes the Send As permission, but the script will export the account as if it did not have Send As permission. This may increase the size of the Export file, but no harm occurs from importing the file and redundantly granting Send As permission to such accounts.
- Active Directory user accounts that have distinguished names and that include tabs or unmatched double quotation marks cannot be processed by using this script. The script can correctly process a name that includes matched double quotation marks such as the following:

  “CN=First “Nickname” Last,DC=domain,DC=com”

- Excel can handle a file that has a maximum of 65,535 lines. If your output file is larger than that, you must split the file into sections before you load it in Excel.
- The Send_As_Errors file will list specific accounts where there was a failure to read or write permissions. If other accounts in the domain were correctly processed, these accounts may have something in common that prevents the script from running with them. Common issues include the following:
  - Lack of administrative permissions to view or to set properties on the accounts.
  - The Exchange mailbox store is not running.
  - The workstation is not a member of the same domain.
  - The administrative account that is being used is not from the same forest.

To run this script, copy and paste all lines between BEGIN SCRIPT and END SCRIPT into a plain text editor such as Notepad. Save the script as AddSendAs.vbs. BEGIN SCRIPT

```
Option Explicit

Dim OUTPUT_DELIMITER
OUTPUT_DELIMITER = """""""" & vbTab & """"""""

'Define exclusive list, if FMA is given to any user in this list, it's ignored.  If you
'want to modify this list, please be sure to follow the same format. Every alias has to
'have a OUTPUT_DELIMITER before and after it
Dim FMA_EXCLUSIVE_LIST
FMA_EXCLUSIVE_LIST = OUTPUT_DELIMITER & "NT AUTHORITY\SELF" & OUTPUT_DELIMITER & "NT
AUTHORITY\SYSTEM" & OUTPUT_DELIMITER
Dim FMA_EXCLUSIVE_EXSVC
FMA_EXCLUSIVE_EXSVC = "\Exchange Services" & OUTPUT_DELIMITER

'Permission Type: Allow or Deny
const ACCESS_ALLOWED_OBJECT_ACE_TYPE  = 5
const ADS_ACETYPE_ACCESS_ALLOWED = &h0
const ADS_ACETYPE_ACCESS_DENIED = &h1

'Flags: Specifies Inheritance
const ADS_ACEFLAG_INHERIT_ACE = &h2
const ADS_ACEFLAG_NO_PROPAGATE_INHERIT_ACE = &h4
const ADS_ACEFLAG_INHERIT_ONLY_ACE = &h8
const ADS_ACEFLAG_INHERITED_ACE = &h10
const ADS_ACEFLAG_VALID_INHERIT_FLAGS = &h1f
const ADS_ACEFLAG_SUCCESSFUL_ACCESS = &h40
const ADS_ACEFLAG_FAILED_ACCESS = &h80

'Declare ADSI constants
Const ADS_OPTION_SECURITY_MASK = 3
Const ADS_OPTION_REFERRALS   = 1
Const ADS_SECURITY_INFO_DACL = 4
Const ADS_CHASE_REFERRALS_NEVER = &h00
Const ADS_CHASE_REFERRALS_SUBORDINATE = &h20
Const ADS_CHASE_REFERRALS_EXTERNAL = &h40

'output file name
Const EXPORT_FILE = "Send_As_Export"
Const ERROR_FILE = "Send_As_Errors"

' script mode
const MODE_INVALID = -1
const MODE_SETALL = 0
const MODE_EXPORT = 1
const MODE_IMPORT = 2
const SETALL = "-SETALL"
const EXPORT = "-EXPORT"
const IMPORT = "-IMPORT"

' argument index
Const ARG_INDEX_MODE = 1
Const ARG_INDEX_DC = 0
Const ARG_INDEX_FILENAME = 2

' column index in import/export file
Const COLUMN_INDEX_USERDISPLAYNAME = 0
Const COLUMN_INDEX_FMAALIAS = 1
Const COLUMN_INDEX_FMADISPLAYNAME = 2
Const COLUMN_INDEX_IFPUBLICDELEGATE = 3
Const COLUMN_INDEX_MAILBOXSTATUS = 4
Const COLUMN_INDEX_USERADSPATH = 5
Const COLUMN_INDEX_HOMEMDB = 6

Const EMPTYSTRING = ""
Const STRNO = "No Delegates"
Const STRYES = "Has Delegates"
Const MIN_ARG = 2
Const INIT_ARRAY_SIZE = 100

' Microsoft Exchange
Const EX_MB_SEND_AS_ACCESSMASK  = &H00100
Const EX_FULLMailbox_AccessMask = 1
Const MESO = "Microsoft Exchange System Objects"
Const EX_MB_SEND_AS_GUID = "{AB721A54-1E2F-11D0-9819-00AA0040529B}"

Const ForReading   = 1
Const ForWriting   = 2
Const ForAppending = 8
Const TristateTrue = -1
Const ADS_SCOPE_SUBTREE = 2

Dim objUser
Dim objSDMailBox
Dim objSDNTsecurity
Dim objDACLNTSD
Dim objNewACE

Dim sTrusteeAlias()
Dim sFMADeniedList
Dim sFMAExplicitAllow
Dim fACESendasFound
Dim dArraySize
Dim TotalACE
Dim i
Dim rootDSE
Dim conn
Dim objCommand
Dim objCmdDisplayName
Dim rsUsers
Dim FoundObject
Dim objFSO
Dim objfileImport
Dim objfileExport
Dim objfileError
Dim sImportFilePath
Dim cScriptMode
Dim dArgCount
Dim dArgExpected
Dim sDCServer
Dim sMailboxStatus
Dim sIfPublicDelegate
Dim sFMAUserDisplayName
Dim sExportFileName
Dim sErrorsFileName
Dim msPublicDelegates
Dim fError
Dim fOneError
Dim fFMAAllowed

On Error Resume Next
dArraySize = INIT_ARRAY_SIZE
ReDim Preserve sTrusteeAlias(dArraySize)

dArgCount = Wscript.Arguments.Count
If ( dArgCount < MIN_ARG ) Then
         DisplaySyntax
End If

err.Clear
fError = False
fOneError = False
cScriptMode = MODE_INVALID
Select Case UCase(WScript.Arguments(ARG_INDEX_MODE))
         Case SETALL
                  cScriptMode = MODE_SETALL
                  dArgExpected = ARG_INDEX_MODE + 1
         Case EXPORT
                  cScriptMode = MODE_EXPORT
                  dArgExpected = ARG_INDEX_MODE + 1
         Case IMPORT
                  cScriptMode = MODE_IMPORT
                  dArgExpected = ARG_INDEX_FILENAME + 1
         Case Else
                  cScriptMode = MODE_INVALID
End Select
If (cScriptMode = MODE_INVALID Or dArgCount <> dArgExpected) Then
         DisplaySyntax
End If

sDCServer = WScript.Arguments(ARG_INDEX_DC)

CreateOutputFiles

If ( cScriptMode = MODE_SETALL Or cScriptMode = MODE_EXPORT ) Then
         Dim sDomainContainer
         If (cScriptMode = MODE_SETALL) Then
                  Dim strInput
                  WScript.StdOut.WriteLine("WARNING: If you continue, each account in the 
domain that has")
                  WScript.StdOut.WriteLine("Full Mailbox Access permission for a given mailbox 
will also be")
                  WScript.StdOut.WriteLine("granted permission to Send As the mailbox owner.")
                  WScript.StdOut.WriteLine()
```

```
                   WScript.StdOut.WriteLine("To preview the list of mailboxes before granting
Send As,")
                   WScript.StdOut.WriteLine("cancel this operation and use the -Export mode of
this script.")
                   WScript.StdOut.WriteLine()
                   WScript.StdOut.Write("Press Y to continue or any other key to cancel: ")
                   strInput = WScript.StdIn.ReadLine()
                   If (UCase(strInput) <> UCase("Y")) Then
                            WScript.Quit
                   End If
          End If

          WScript.StdOut.WriteLine()
          WScript.StdOut.WriteLine("""!"" indicates an error processing an object.")
          WScript.StdOut.WriteLine("     Check " & sErrorsFilename)
          WScript.StdOut.WriteLine("Starting...")
          WScript.StdOut.WriteLine()

          err.Clear
          Set rootDSE = GetObject("LDAP://" & sDCServer & "/RootDSE")
          sDomainContainer = rootDSE.Get("defaultNamingContext")
          WScript.StdOut.WriteLine("Finding domain controller [ " & sDCServer & " ] for domain
[ " & sDomainContainer & " ]")

          If (err.number <> 0) Then
                   WScript.StdOut.WriteLine("Failed to find the domain or domain controller,
error:" & err.Description)
                   objfileError.WriteLine("Failed to find the domain or domain controller,
error:" & err.Description)
                   WScript.Quit
          End If

          err.Clear
          Set conn = CreateObject("ADODB.Connection")
          Set objCommand = CreateObject("ADODB.Command")
          conn.Provider = "ADSDSOObject"
          conn.Open "ADs Provider"
          If (err.number <> 0) Then
                   WScript.StdOut.WriteLine("Failed to bind to Active Directory server, error:"
& err.Description)
                   objfileError.WriteLine("Failed to bind to Active Directory server, error:" &
err.Description)
                   WScript.Quit
          End If

          Set objCommand.ActiveConnection = conn
          WScript.StdOut.WriteLine("Searching for mailbox owner user accounts in " &
sDomainContainer)

          objCommand.CommandText  = "<LDAP://" & sDCServer & "/" & sDomainContainer & ">;(&(&(&
(mailnickname=*) (| (&(objectCategory=person)(objectClass=user)
(msExchHomeServerName=*)) ))));adspath;subtree"
          objCommand.Properties("searchscope") = ADS_SCOPE_SUBTREE
          objCommand.Properties("Page Size") = 100
          objCommand.Properties("Timeout") = 30
          objCommand.Properties("Chase referrals") = (ADS_CHASE_REFERRALS_SUBORDINATE Or
ADS_CHASE_REFERRALS_EXTERNAL)

          err.Clear
          Set rsUsers = objCommand.Execute
          If (err.number <> 0) Then
                   WScript.StdOut.WriteLine("Search for mailbox owners failed, error:" &
err.Description)
                   objfileError.WriteLine("Search for mailbox owners failed, error:" &
err.Description)
                   WScript.Quit
          End If

          If (rsUsers.RecordCount = 0) Then
                   WScript.StdOut.WriteLine("No mailbox owner user accounts could be seen in " &
sDomainContainer & ".")
```

```
                    objfileError.WriteLine("No mailbox owner user accounts found in " & 
sDomainContainer & ".")
                    fError = True
          End If

          While Not rsUsers.EOF
                    If (fOneError = True) Then
                              WScript.StdOut.Write("!")
                    Else
                              WScript.StdOut.Write(".")
                    End If
                    fOneError = False

                    'Skip any mailbox object in Microsoft Exchange System Objects container
                    If (0 = Instr(rsUsers.Fields(0).Value, MESO)) Then
                              err.Clear
                              Set objUser = GetObject(rsUsers.Fields(0).Value)
                              If (err.number <> 0) Then
                                        objfileError.WriteLine("Failed to get user object: " & 
rsUsers.Fields(0).Value)
                                        objfileError.WriteLine("Error: " & err.Description)
                                        fError = True
                                        fOneError = True
                                        err.Clear
                              End If
                              Set objSDMailBox = objUser.MailboxRights
                              If (err.number <> 0) Then
                                        objfileError.WriteLine("Failed to get mailbox rights: " & 
rsUsers.Fields(0).Value)
                                        objfileError.WriteLine("Error: " & err.Description)
                                        fError = True
                                        fOneError = True
                                        err.Clear
                              End If
                              Set objSDNTsecurity = objUser.ntSecurityDescriptor
                              If (err.number <> 0) Then
                                        objfileError.WriteLine("Failed to get NTSD: " & 
rsUsers.Fields(0).Value)
                                        objfileError.WriteLine("Error: " & err.Description)
                                        fError = True
                                        fOneError = True
                                        err.Clear
                              End If

                              Set objDACLNTSD = Nothing
                              If (objUser.AccountDisabled) Then
                                        sMailboxStatus = "Disabled"
                              Else
                                        sMailboxStatus = "Enabled"
                              End If

                              'Query this user's publicDelegates list
                              err.Clear
                              msPublicDelegates = objUser.Get("publicDelegates")
                              If (err.number <> 0) Then
                                        'This user doesn't have publicDelegates list set
                                        sIfPublicDelegate = STRNO
                                        err.Clear
                              Else
                                        sIfPublicDelegate = STRYES
                              End If

                              err.Clear
                              FindAllFMAUsers objSDMailBox

                              If (TotalACE > dArraySize) Then
                              'Needs to allocate bigger size array
                                        dArraySize = TotalACE + 1
                                        ReDim Preserve sTrusteeAlias(dArraySize)
                                        FindAllFMAUsers objSDMailBox
                              End If
```

```
                                If (err.number <> 0) Then
                                        objfileError.WriteLine("Failed to query mailbox rights of
user: " & rsUsers.Fields(0).Value)
                                        objfileError.WriteLine("Error: " & err.Description)
                                        err.Clear
                                        fError = True
                                        fOneError = True
                                End If

                                If TotalACE > 0 Then
                                        Set objDACLNTSD = objSDNTsecurity.DiscretionaryAcl

                                        For i = 0 to TotalACE - 1 Step 1

                                                'Check if we already have Send As ACE in NT
security descriptor
                                                'If it exists, either allow or deny, we don't
need to add send as to it
                                                CheckSendAsACE objDACLNTSD, sTrusteeAlias(i)

                                                'Note: deny entries take precedence over allow
entries.
                                                'If there is FMA deny ACE, skip it even if we
find FMA allow ACE
                                                IfFMAAllowed(sTrusteeAlias(i) & OUTPUT_DELIMITER)
                                                If ((fFMAAllowed = True) And (fACESendasFound =
0)) Then
                                                        If cScriptMode = MODE_SETALL Then
                                                                Set objNewACE = CreateObject
("AccessControlEntry")
                                                                objNewACE.AceFlags = 0
                                                                objNewACE.AceType =
ACCESS_ALLOWED_OBJECT_ACE_TYPE
                                                                objNewACE.AccessMask =
EX_MB_SEND_AS_ACCESSMASK
                                                                objNewACE.Flags = 1
                                                                objNewACE.ObjectType =
EX_MB_SEND_AS_GUID
                                                                objNewACE.Trustee =
sTrusteeAlias(i)

                                                                objDACLNTSD.AddAce objNewAce
                                                        End If

                                                        'Query trustee(FMA user) to get its
displayName
                                                        Dim rsTrustee
                                                        Dim objTrustee
                                                        Dim dPosition
                                                        Dim sAlias

                                                        dPosition = inStr(1, sTrusteeAlias
(i), "\")
                                                        sAlias = mid(sTrusteeAlias(i),
dPosition + 1)

                                                        Set objCmdDisplayName = CreateObject
("ADODB.Command")
                                                        Set objCmdDisplayName.ActiveConnection
= conn
                                                        objCmdDisplayName.CommandText
= "<LDAP://" & sDomainContainer & ">;(&(&(& (mailnickname=" & sAlias & ") (| (&
(objectCategory=person)(objectClass=user)(msExchHomeServerName=*)) ))));adspath;subtree"
                                                        objCmdDisplayName.Properties
("searchscope") = ADS_SCOPE_SUBTREE
                                                        objCmdDisplayName.Properties("Page
Size") = 100
                                                        objCmdDisplayName.Properties("Timeout")
= 30
                                                        objCmdDisplayName.Properties("Chase
referrals") = (ADS_CHASE_REFERRALS_SUBORDINATE Or ADS_CHASE_REFERRALS_EXTERNAL)
```

```
                                                          Set rsTrustee = 
objCmdDisplayName.Execute
                                                          Set objTrustee = GetObject
(rsTrustee.Fields(0).Value)

                                                          If (err.number <> 0) Then
                                                                  'Failed to query FMA user's
display name, use its alias
                                                                  sFMAUserDisplayName = sAlias

                                                          Else
                                                                  sFMAUserDisplayName = 
objTrustee.displayName
                                                          End If

                                                          'output to export file
                                                          err.Clear
                                                          objfileExport.WriteLine ("""""""""" & 
objUser.displayName & OUTPUT_DELIMITER & sTrusteeAlias(i) & OUTPUT_DELIMITER & 
sFMAUserDisplayName & OUTPUT_DELIMITER & sIfPublicDelegate & OUTPUT_DELIMITER & sMailboxStatus & 
OUTPUT_DELIMITER & rsUsers.Fields(0).Value & OUTPUT_DELIMITER & objUser.homeMDB & """""""""")
                                                          If (err.number <> 0) Then
                                                                  objfileError.WriteLine 
("User " & rsUsers.Fields(0).Value & " could not be added to the export file. You should set 
permissions manually for this user.")
                                                                  objfileError.WriteLine 
("Error: " & err.Description)

                                                                  err.Clear
                                                                  fError = True
                                                                  fOneError = True
                                                          End If
                                                          Set objCmdDisplayName = Nothing
                                                          Set rsTrustee = Nothing
                                                          Set objTrustee = Nothing
                                                  End If
                                          Next

                                          If cScriptMode = MODE_SETALL Then
                                                  err.Clear
                                                  objSDNTsecurity.DiscretionaryAcl = objDACLNTSD
                                                  objUser.Put "ntSecurityDescriptor", Array( 
objSDNTsecurity )
                                                  objUser.SetOption ADS_OPTION_SECURITY_MASK, 
ADS_SECURITY_INFO_DACL
                                                  objUser.SetInfo
                                                  If (err.number <> 0) Then
                                                          objfileError.WriteLine("Failed to 
update ADSI for user: " & rsUsers.Fields(0).Value)
                                                          objfileError.WriteLine("Error: " & 
err.Description)
                                                          err.Clear
                                                          fError = True
                                                          fOneError = True
                                                  End If
                                          End If

                                          TotalACE = 0
                                          Set objSDMailbox = Nothing
                                          Set objSDNTsecurity = Nothing
                                          Set objUser = Nothing
                                          Set objDACLNTSD = Nothing
                                  End If

                          End If
                          rsUsers.MoveNext
                  Wend
          End If

          If (cScriptMode = MODE_IMPORT) Then
                  Dim sOneRow
```

```
         Dim sArraySplit
         Dim objUserItem
         Dim UserPath
         Dim objUserSD
         Dim objUserDACL
         Dim fNeedToAddSendAs

         sImportFilePath = WScript.Arguments(ARG_INDEX_FILENAME)

         WScript.StdOut.WriteLine("If you continue, each account listed in " & sImportFilePath)
         WScript.StdOut.WriteLine("that has Full Mailbox Access permission for a given mailbox")
         WScript.StdOut.WriteLine("will also be granted permission to Send As the mailbox
owner.")
         WScript.StdOut.WriteLine()
         WScript.StdOut.Write("Press Y to continue or any other key to cancel: ")
         strInput = WScript.StdIn.ReadLine()
         If (UCase(strInput) <> UCase("Y")) Then
                  WScript.Quit
         End If
         WScript.StdOut.WriteLine("Starting...")
         WScript.StdOut.WriteLine()

         UserPath = EMPTYSTRING
         err.Clear
         Set objFSO = CreateObject("Scripting.FileSystemObject")
         Set objfileImport = objFSO.OpenTextFile(sImportFilePath, ForReading, False,
TristateTrue)
         If (err.number <> 0) Then
                  WScript.StdOut.WriteLine("Failed to open import file " & sImportFilePath & ",
error:" & err.Description)
                  objfileError.WriteLine("Failed to open import file " & sImportFilePath & ",
error:" & err.Description)
                  WScript.Quit
         End If

         fNeedToAddSendAs = False
         Do While objfileImport.AtEndOfStream <> True
                  If (fOneError = True) Then
                           WScript.StdOut.Write("!")
                  Else
                           WScript.StdOut.Write(".")
                  End If
                  fOneError = False

                  err.Clear
                  sOneRow = objfileImport.ReadLine
                  sArraySplit = Split(sOneRow , OUTPUT_DELIMITER)
                  If (err.number <> 0) Then
                           objfileError.WriteLine("Failed to parse one row: " & sOneRow )
                           objfileError.WriteLine("Error: " & err.Description)
                           err.Clear
                           fError = True
                           fOneError = True
                  End If

                  If (UserPath <> sArraySplit(COLUMN_INDEX_USERADSPATH)) Then
                           'A new user
                           If (fNeedToAddSendAs = True ) Then
                                    'update existing user
                                    err.Clear
                                    objSDNTsecurity.DiscretionaryAcl = objDACLNTSD
                                    objUser.Put "ntSecurityDescriptor", Array(
objSDNTsecurity )
                                    objUser.SetOption ADS_OPTION_SECURITY_MASK,
ADS_SECURITY_INFO_DACL
                                    objUser.SetInfo
                                    If (err.number <> 0) Then
                                             objfileError.WriteLine("Failed to update
permissions for user: " & UserPath)
                                             objfileError.WriteLine("Error: " &
err.Description)
```

```
                                                fError = True
                                                fOneError = True
                                        End If
                                End If

                                fNeedToAddSendAs = False
                                Set objUser = Nothing
                                Set objSDNTsecurity = Nothing
                                Set objDACLNTSD = Nothing

                                UserPath = sArraySplit(COLUMN_INDEX_USERADSPATH)
                                err.Clear
                                Set objUser = GetObject(UserPath)
                                Set objSDNTsecurity = objUser.ntSecurityDescriptor
                                Set objDACLNTSD = objSDNTsecurity.DiscretionaryACL

                                If (err.number <> 0) Then
                                        objfileError.WriteLine("Failed to get user object: " &
UserPath)
                                        objfileError.WriteLine("Error: " & err.Description)
                                        err.Clear
                                        fError = True
                                        fOneError = True
                                End If
                        End If

                        'Add newACE   Do we need this check?
                        CheckSendAsACE objDACLNTSD, sArraySplit(COLUMN_INDEX_FMAALIAS)
                        If (fACESendasFound = 0) Then
                                Set objNewACE = CreateObject ("AccessControlEntry")
                                objNewACE.AceFlags = 0
                                objNewACE.AceType = ACCESS_ALLOWED_OBJECT_ACE_TYPE
                                objNewACE.AccessMask = EX_MB_SEND_AS_ACCESSMASK
                                objNewACE.Flags = 1
                                objNewACE.ObjectType = EX_MB_SEND_AS_GUID
                                objNewACE.Trustee = sArraySplit(COLUMN_INDEX_FMAALIAS)

                                objDACLNTSD.AddAce objNewACE
                                fNeedToAddSendAs = True
                        End If
                Loop

                If (fNeedToAddSendAs = True ) Then
                        'update the last user
                        err.Clear
                        objSDNTsecurity.DiscretionaryAcl = objDACLNTSD
                        objUser.Put "ntSecurityDescriptor", Array( objSDNTsecurity )
                        objUser.SetOption ADS_OPTION_SECURITY_MASK, ADS_SECURITY_INFO_DACL
                        objUser.SetInfo
                        If (err.number <> 0) Then
                                objfileError.WriteLine("Failed to update permissions for user: " &
UserPath)
                                objfileError.WriteLine("Error: " & err.Description)
                                fError = True
                        End If
                End If

End If

objFSO.Close
objfileImport.Close
objfileExport.Close
objfileError.Close

Set objFSO = Nothing
Set objfileImport = Nothing
Set objfileExport = Nothing
Set objfileError = Nothing
Set objCommand = Nothing
Set conn = Nothing
```

```
WScript.StdOut.WriteLine()
If (fError = True) Then
         WScript.StdOut.WriteLine("Finished with one or more errors. See " & sErrorsFilename)
Else
         WScript.StdOut.WriteLine("Finished successfully. No errors were encountered.")
End If

Function FindAllFMAUsers (objSD)
Dim objACL
Dim objACE
Dim intACECount
Dim strIndent
Dim dAccessMaskBit
Dim dPosition
Dim sUserAlreadyFound

         On Error Resume Next
         err.Clear
         TotalACE = 0
         sFMADeniedList = EMPTYSTRING
         sFMAExplicitAllow = EMPTYSTRING
         sUserAlreadyFound = OUTPUT_DELIMITER
         intACECount = 0
         Set objACL = objSD.DiscretionaryAcl
         intACECount = objACL.AceCount

         If intACECount Then
                  ' Open discretionary ACL (DACL) data.
                  For Each objACE In objACL

                  dPosition = inStr(1, objACE.Trustee, "$")
                  If ((0 = Instr(UCase(objACE.Trustee & OUTPUT_DELIMITER), UCase
(FMA_EXCLUSIVE_EXSVC))) And (0 = Instr(sUserAlreadyFound, OUTPUT_DELIMITER & objACE.Trustee &
OUTPUT_DELIMITER)) And (0 = Instr(FMA_EXCLUSIVE_LIST, OUTPUT_DELIMITER & objACE.Trustee &
OUTPUT_DELIMITER)) And (dPosition <> Len(objACE.Trustee)) And ((objACE.AccessMask And
EX_FULLMailbox_AccessMask) <>0) And ((objACE.AceType = ADS_ACETYPE_ACCESS_ALLOWED) Or
(objACE.AceType = ACCESS_ALLOWED_OBJECT_ACE_TYPE) )) Then
                           If (TotalACE < dArraySize) Then
                                    sTrusteeAlias(TotalACE) = objACE.Trustee
                                    sUserAlreadyFound = sUserAlreadyFound & objACE.Trustee &
OUTPUT_DELIMITER
                           End If
                           TotalACE = TotalACE + 1
                           If ((objACE.AceFlags And ADS_ACEFLAG_INHERITED_ACE) = 0) Then
                                    'Keep a list who explictly set FMA at mailbox level
                                    sFMAExplicitAllow = sFMAExplicitAllow & objACE.Trustee &
OUTPUT_DELIMITER
                           End If
                  ElseIf (( (objACE.AccessMask And EX_FULLMailbox_AccessMask) <>0 ) And
(objACE.AceType = ADS_ACETYPE_ACCESS_DENIED)) Then
                           'Keep a list who denied FMA, use OUTPUT_DELIMITER as demiliter,
                           'include both inherited and explicit set at mailbox level
                           sFMADeniedList = sFMADeniedList & objACE.Trustee & OUTPUT_DELIMITER

                  End If
                  Next
         End If

         Set objACL = Nothing
End Function

Function CheckSendAsACE (objDiscretionaryACL, sTAlias)
Dim objACE
Dim intACECount

         err.Clear
         fACESendasFound = 0
         intACECount = objDiscretionaryACL.AceCount

         If intACECount Then
                  For Each objACE In objDiscretionaryACL
```

```
                              err.Clear
                              If ( (objACE.Trustee = sTAlias) And (objACE.ObjectType =
EX_MB_SEND_AS_GUID) ) Then
                                        fACESendasFound = 1
                              End If
                              If (err.number <> 0) Then
                                        objfileError.WriteLine("Could not read permissions for
this user: " & sTAlias)
                                        objfileError.WriteLine("Error: " & err.Description)
                                        err.Clear
                                        fError = True
                                        fOneError = True
                              End If
                    Next
          End If
End Function

Function IfFMAAllowed(sTrustee)
          'FMA allow ACE has been found. Assume it's true
          fFMAAllowed = True

          If ( (0 <> Instr(sFMADeniedList, sTrustee)) And (0 = Instr(sFMAExplicitAllow, sTrustee))
          ) Then
                    'If Denied ACE is found, and no explicit allow FMA
                    fFMAAllowed = False
          End If
End Function

Function CreateOutputFiles
          Dim sTimeArray
          Dim sTimeShort
          Dim sTime

          err.Clear
          sTime = Time
          sTimeShort = Split(sTime, " ")
          sTimeArray = Split(sTimeShort(0), ":")

          Set objFSO = CreateObject("Scripting.FileSystemObject")
          sErrorsFileName = ERROR_FILE & "_" & sTimeArray(0) & "_" & sTimeArray(1) & "_" &
sTimeArray(2) & ".txt"
          Set objfileError = objFSO.OpenTextFile(sErrorsFileName, ForWriting, True, TristateTrue)

          If (cScriptMode = MODE_SETALL Or cScriptMode = MODE_EXPORT)        Then
                    sExportFileName = EXPORT_FILE & "_" & sTimeArray(0) & "_" & sTimeArray(1)
& "_" & sTimeArray(2) & ".txt"
                    Set objfileExport = objFSO.OpenTextFile(sExportFileName, ForWriting, True,
TristateTrue)
          End If

          If err.number <> 0 Then
                    WScript.StdOut.WriteLine("Unable to create export or error files: " &
err.Description)
                    objfileError.WriteLine("Unable to create export or error files: " &
err.Description)
                    fError = True
                    fOneError = True
                    WScript.Quit
          End If

End Function

Function DisplaySyntax
          WScript.StdOut.WriteLine("Syntax:")
          WScript.StdOut.WriteLine()
          WScript.StdOut.WriteLine("Export accounts with Full Mailbox Access that do not have
Send As permission:")
          WScript.StdOut.WriteLine("     CSCRIPT """ & WScript.ScriptName & """
DOMAIN_CONTROLLER -Export")
          WScript.StdOut.WriteLine("         NOTE: The list will be saved to
Send_As_Export_HH_MM_SS.txt")
```

```
          WScript.StdOut.WriteLine()
          WScript.StdOut.WriteLine("Grant Send As to all accounts listed in an export file:")
          WScript.StdOut.WriteLine("     CSCRIPT """ & WScript.ScriptName & """
DOMAIN_CONTROLLER -Import ""filename.txt""")
          WScript.StdOut.WriteLine()
          WScript.StdOut.WriteLine("Grant Send As to all accounts in the domain with Full Mailbox
Access:")
          WScript.StdOut.WriteLine("     CSCRIPT """ & WScript.ScriptName & """
DOMAIN_CONTROLLER -SetAll")
          WScript.StdOut.WriteLine("         NOTE: Accounts will be listed in
Send_As_Export_HH_MM_SS.txt")
          WScript.StdOut.WriteLine()
          WScript.StdOut.WriteLine("For all modes, errors are saved to
Send_As_Errors_HH_MM_SS.txt")

          WScript.Quit
End Function
```

END SCRIPT

For additional information about the support options that are available from Microsoft, visit the following Microsoft Web site:

http://support.microsoft.com/default.aspx?scid=fh;[LN];CNTACTMS (http://support.microsoft.com/default.aspx?scid=fh;%5Bln%5D;cntactms)

APPLIES TO

- Microsoft Exchange 2000 Server Standard Edition
- Microsoft Exchange 2000 Enterprise Server
- Microsoft Exchange Server 2003 Standard Edition
- Microsoft Exchange Server 2003 Enterprise Edition

Keywords: kbtshoot kbpending kbbug kbprb KB912918