Sun Java™ Enterprise System Technical Note: Configuring Sun Java Messaging Server MTA to Control Sending Email on Behalf of Another (Delegate Access)

2005Q1

Part Number 819-4357-10

The *Sun Java Enterprise System 2005Q1 Technical Note: Configuring Sun Java Messaging Server MTA to Control Sending Email on Behalf of Another (Delegate Access)* describes how to configure your Sun Java System Messaging Server environment such that one user can send email on behalf of another user. In Microsoft Exchange terms, this capability is referred to as Delegate Access.

The component products affected by this technical note are:

- Sun Java System Messaging Server 6 2005Q1
- Sun Java System Directory Server 5 2005Q1

This technical note contain the following sections:

Technical Note Revision History

Table 1 Revision History

Date	Description of Changes
October 7, 2005	Minor formatting changes to the AUTH_REWRITE table.
October 5, 2005	Initial release of this technical note.

Overview of This Technical Note

Currently, products such as Microsoft Exchange, and other groupware systems, allow for the setting of permissions such that one user can be configured to send email on behalf of another user. For example, just as managers might have assistants who help manage their paper mail, these managers can use Microsoft Exchange to give another person access to their email. In Microsoft Exchange, the process of granting someone permission to open folders, read and create items, and respond to requests for another person is called Delegate Access.

This means that customers who are considering migrating their email infrastructure from Exchange to Messaging Server most likely require similar Delegate Access functionality in Sun Java System Messaging Server. This document describes how to configure the LDAP directory and the Messaging Server MTA to provide analogous functionality that exists in Microsoft Exchange.

Any email or groupware system that enables you to take actions based upon the identity of the sending user requires that you know who the actual sender of that email is. Typically, when an email originates from “inside” (in MTA configuration terms, this means the sender’s IP address specified in the INTERNAL_IP mapping), the submission of the message is allowed regardless of its destination and claimed sender. This is the MTA’s default configuration. But to be able to successfully apply mail submission delegation, you need to control who is sending email, even if that mail comes from inside. Such a situation leads to authentication enforcement even for these messages.

Using the sender’s address—either the email envelope or header—is not suitable. Envelope and header addresses can easily be forged. Because these addresses are easily forged, organizations should not rely upon them, especially when they want their users to be able to send sensitive email, often on behalf of company managers.

Controlling associated authentication information is usually the most feasible way of being certain who the actual sender of a particular email message is. Such information is available only if the sender of that message authenticates (uses a password or other authentication mechanism) upon message submission. This document describes a way to be able to perform decisions based on authentication information by enforcing authentication of all email in the environment. The only exceptions would be for email arriving from the Internet, and some email that is sent automatically and thus can be easily recognized. In this document, authenticating all email in an organization's environment means all email that users send to one another within the organization, as well all email that users send outside the organization.

Configuring the LDAP Directory and MTA

The capability to verify the permission of who can send email on behalf of whom needs to be performed in several steps. In the technique used here, the data of who can send email on behalf of another is stored in the LDAP directory server. Once you configure the LDAP directory to accept this data, you then configure the MTA through the use of a mapping file to check permissions.

Creating the configuration includes three basic operations:

1. Adding a new attribute that stores information about who can send mail on behalf of a particular user, to the directory server's schema
2. Adding a new objectclass and values for the attribute to the user's entry
3. Configuring the MTA through the use of a mapping file to check for permissions

This document does not discuss how to enforce authentication of internal-to-internal and internal-to-outside email, as that is part of the normal MTA configuration (though such a configuration is probably not very usual). Refer to the *Sun Java System Messaging Server 6 2005Q1 Administration Guide* for the description of such a configuration.

Provisioning Data into the LDAP Directory

Throughout this document, the person who permits submission is called a *manager*, and the person that these permissions are granted to is called an *assistant*. In the technique used here, the data of who can send email on behalf of whom is stored in the directory server, in particular in the entry of the manager. For the sake of this discussion, these permissions are stored in an attribute called `mailGrantSendPermissionsTo`. Such an attribute does not exist in any of the Sun-distributed schemas. You need to add this attribute to the configuration of your directory server.

➤ To Add a New Attribute to the LDAP Directory

- One way to add this attribute is by creating a file called `99grant.ldif`, with the following content, and placing it in the "schema" directory of the directory server:

```
dn: cn=schema
attributeTypes: ( mailGrantSendPermissionsTo-oid NAME ( 'mailGrant
 SendPermissionsTo' ) DESC 'Attribute for granting send permissio
 ns' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'grant permissions' )
objectClasses: ( mailGrantPermission-oid NAME 'mailGrantPermissi
 on' DESC 'An objectclass for storing send permissions' SUP inetLoc
 alMailRecipient MAY ( mailGrantSendPermissionsTo ) X-ORIGIN 'gra
 nt permissions' )
```

The default Java Enterprise System location for the "schema" directory is `/var/opt/mps/serverroot/slapd-instance/config/schema`.

➤ To Add a New Objectclass to the LDAP Directory

- After adding the objectclass and the attribute to your LDAP directory server schema, you are able to add the `mailGrantPermission` objectclass to the user's entry. Along with that, you can add a number of values for the `mailGrantSendPermissionTo` attribute. You should add that data to the manager's LDAP entry, and the `mailGrantSendPermissionTo` values should contain mail addresses of assistants of that manager (one address per value).[1]

  For example, to add permission so that a user whose email address is `assistant@red.example.com` can send mail on behalf of a user whose DN is `uid=manager, ou=People, o=red.example.com, dc=red, dc=example, dc=com`, you need to add the following attributes to the latter, as shown by this ldif file example.

```
dn: uid=manager,ou=People,o=red.example.com,dc=red,dc=example,dc=com
changetype: modify
add: objectclass
objectClass: mailGrantPermission
-
add: mailGrantSendPermissionsTo
mailGrantSendPermissionsTo: assistant@red.example.com
```

You would use the `ldapmodify` command with this file to make the changes in the directory.

1.The reason for using mail address rather than a DN is that this will later be verified by Messaging Server, and it has access to authentication information in the form of a mail address, thus easing the MTA configuration.

Configuring the MTA to Check the Permissions

For the next part of this discussion, you need to be familiar with the SMTP protocol and how an email message is organized and submitted to the server.

When Outlook sends an email to the server using the SMTP protocol, and the author of that email (an assistant) sends it on behalf of somebody else (a manager), and SMTP authentication is enabled on the client and allowed on the server, that mail has following characteristics:

- The authentication information is the assistant
- The envelope from information (the address which is used in the `MAIL FROM:` command during the mail submission) is the assistant
- The header `From:` address is the manager
- The header `Sender:` field is the assistant

➤ To Configure the MTA to Check for Permissions

With this information in mind, you need to configure the Messaging Server MTA as follows:

1. Add `autherwrite 3` to the definitions of those channels for which you want to enable the check, in the `imta.cnf` file. This can be, for instance, the `tcp_auth` channel, if the messages from authenticated users are switched to that channel.
2. Add the following lines to the mapping file:

```
AUTH_REWRITE                                                                    (0)

    *|*|*|$2* $Y$2                                                              (1)
    *|*@*|*@* $CBASE|$}$4,_base_dn_{|$1@$2|$3@$4                                (2)
    BASE|*|*|* $CFOUND|$]ldap:///$0?uid?sub?(&(mail=$2)                         (3)
          (|(mailAlternateAddress=$1)(mailEquivalentAddress=$1)))[|$2
    FOUND|*|* $Y$1                                                              (4)
    BASE|*|*@*|*@* $CSECONDARY_BASE|$}$2,_base_dn_{|$1@$2|$3@$4                 (5)
    SECONDARY_BASE|*|*|* $CSECONDARY_FOUND|$]ldap:///$0?uid?sub?(&(|(mail=$1)   (6)
           mailAlternateAddress=$1)(mailEquivalentAddress=$1))
          (mailGrantSendPermissionsTo=$2))[|$2
    SECONDARY_FOUND|*|* $Y$1                                                    (7)
    * $NYou$ have$ no$ permission$ to$ send$ mail$ on$ behalf$ of$ this$ person (8)
```

NOTE The numbers at the end of the lines are used for the discussion that follows. The lines (3) and the next one, and the line (6) and two following lines, should really be a single line, but they have been broken for readability.

Understanding How the Configuration Works

If you place the `authrewrite 3` keyword on the channel that an email message comes through, that email goes through the `AUTH_REWRITE` mapping table. Consequently, the server creates the following probe:

```
mail-from|sender|from|auth-sender
```

where `mail-from` is the envelope from address, `sender` is the address from the `Sender:` or `Resent-sender:` header field, `from` is the address from the `From:` or `Resent-From:` header field, and `auth-sender` is the address provided by the authentication operation. The result is run through the `AUTH_REWRITE` mapping. That mapping is supposed to return a result that indicates what the server's behavior regarding this message should be. The result can contain (but is not limited to) the following items: `$N` (reject the message) and `$Y` (add an appropriate `Sender:` header field, which also implies accepting the message).[2]

Thus, if `assistant@red.example.com` tries to send email on behalf of `manager@red.example.com` with Outlook, and is authenticated, the probe for `AUTH_REWRITE` is constructed in a following way:

```
assistant@red.example.com|assistant@red.example.com|
  manager@red.example.com|assistant@red.example.com
```

This is a single line (broken here for readability) and it is tested in the following way by the mapping file:

1. Line (1) checks if the `From:` header field is the same as the authentication address. This is likely to be the case in vast majority of messages, in particular in all the messages that are sent when an email is not sent on behalf of anybody, just the sender information is used everywhere.

2.The AUTH REWRITE mapping, as well as information about applying mappings in general, can be found in the *Sun Java System Messaging Server 2005Q1 Administration Guide.*

2. Line (2) finds the base DN to perform lookups for the assistant LDAP entry, based on the domain that is used in the authentication address. If the base DN is found (and actually the DN is always found, as the domain from the authentication information must be local), the probe that is later checked is changed to:

   `BASE|dn|from|auth-sender`

   where `dn` is the base DN that is found. In our case it can be:

   ```
   BASE|o=red.example.com,dc=red,dc=example,dc=com|
   manager@red.example.com|assistant@red.example.com
   ```

   (Note: This is a single line, broken for readability.)

3. Line (3) checks if the probe starts from `BASE|`. If so, it means that the previous search succeeded (which must be the case, as explained before). In this case, the probe checks if the address being sent from is one of the sender's own. This is achieved by looking at which mail address (the mail attribute) is the sender's authenticated address, and if the header `From:` address is one of the same entry's aliases (`mailAlternateAddress` and `mailEquivalentAddress` attributes are checked). In case that an entry is found, the probe is changed so it starts with `FOUND|`, otherwise the probe remains as it was before the lookup.

4. If the probe starts from `FOUND|`, the mail is accepted, and the `Sender:` line containing the authenticated address is added to its header. This is done in line (4).

5. Otherwise, the probe tries to find the base DN for the manager's entry. The reason for this is to support sending on behalf of users in different domains. This search may or may not succeed, depending on whether that address is local or not. As this technique does not intend to support sending on behalf on non-local users, if the base DN for user search fails, the email will be later rejected (in line (8)). If the base DN is found, the probe is changed so it starts from the `SECONDARY_BASE` string. In this example, the probe becomes:

   ```
   SECONDARY_BASE|o=red.example.com,dc=red,dc=example,dc=com|
   manager@red.example.com|assistantw@red.example.com
   ```

   (Note: This is a single line, broken for readability.) The base DN in the example is the same as in step 2, as both the assistant and the manager belong to the same domain, but it might very well be different.

6. If the base DN is found, then the manager entry is found, and it is checked if the send permissions are granted to the assistant. This is achieved with a single search (line (6)), that looks for a single entry that has any of the values of `mail`, `mailAlternateAddress`, or `mailEquivalentAddress` addresses equal to the header `From:` address, and one of the values of `mailGrantSendPermissionsTo` is equal to the authenticated sender's address. If such an entry is found, the probe is changed so it begins from `SECONDARY_FOUND`.

   As this technique does not intend to support sending on behalf on non-local users, if the base DN for user search fails, the email will be later rejected (in line (8)). If you want to allow users to send on behalf of non-local addresses, alter the mapping as described in "Allowing Users to Send on Behalf of Non-local Addresses."

7. If the probe starts from `SECONDARY_FOUND`, the mail is accepted (line (7)).

8. Otherwise the mail is rejected (line (8)). The rejection applies also to the situation when a base DN is not found for the manager's domain (that is, the manager's address in not local, this is checked in line (5)).

Each time when an email is accepted for delivery, the header `Sender:` domain is set to the value of the authenticated address.

The rejection of email happens during the submission, as a response to the `.` (the final dot). The treatment of the rejection depends on the mail user agent that is used.

Allowing Users to Send on Behalf of Non-local Addresses

If you want to allow internal users send on behalf of external addresses, add the following line between lines (7) and (8) in the previous code example:

```
BASE|*|*|* $Y$2
```

Adding this line makes the MTA accept any email sent by an internal user, in which header `From:` address is not from a locally hosted domain. At the same time, adding this line still allows for checking permission to send on behalf of other local users. No check is made on whether a local user can send using a particular external address in the header `From:` field.

Further Reading

See the *Sun Java System Messaging Server 2005Q1 Administration Guide* for more details:

`http://docs.sun.com/source/819-0105`

Known Issues and Limitations

See the Java Enterprise System Release Notes Collection at the following URL to find out about known problems:

http://docs.sun.com/app/docs/coll/entsysrn_05q1

How to Report Problems and Provide Feedback

If you have problems with Communications Express, contact Sun customer support using one of the following mechanisms:

- Sun Software Support services online at
  http://www.sun.com/service/sunone/software

  This site has links to the Knowledge Base, Online Support Center, and ProductTracker, as well as to maintenance programs and support contact numbers.
- The telephone dispatch number associated with your maintenance contract

So that we can best assist you in resolving problems, please have the following information available when you contact support:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to http://docs.sun.com and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Communications Express 2005Q1 Technical Note: Configuring Sun Java Messaging Server MTA to Control Sending Email on Behalf of Another (Delegate Access)*, and the part number is 819-4357-10.

Additional Sun Resources

Useful Sun Java System information can be found at the following Internet locations:

- Sun Java System Documentation
  http://docs.sun.com/prod/java.sys
- Sun Java System Professional Services
  http://www.sun.com/service/sunps/sunone
- Sun Java System Software Products and Service
  http://www.sun.com/software
- Sun Java System Software Support Services
  http://www.sun.com/service/sunone/software
- Sun Java System Support and Knowledge Base
  http://www.sun.com/service/support/software
- Sun Support and Training Services
  http://training.sun.com
- Sun Java System Consulting and Professional Services
  http://www.sun.com/service/sunps/sunone
- Sun Java System Developer Information
  http://developers.sun.com
- Sun Developer Support Services
  http://www.sun.com/developers/support
- Sun Java System Software Training
  http://www.sun.com/software/training
- Sun Software Data Sheets
  http://wwws.sun.com/software