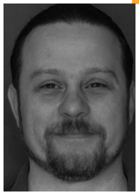
TROY MCKEE

migrating from hosted Exchange service to in-house Exchange solution



Troy McKee is an IT manager with an eclectic range of experience, from geology and engineering to the games industry and software development. Having worked in just about every sized environment and on many platforms, he currently prefers to work wherever he finds a challenge. Troy has a master's degree in IT from American Intercontinental University. He is currently employed in the financial services industry, which is quite challenging.

tmckee@leadclick.com

IN THIS ARTICLE I REVIEW SOME OF the issues involved in migrating from a hosted Exchange service to an in-house Exchange solution. I will briefly discuss the choice to migrate, the pros and cons of an in-house Exchange solution versus a hosted Exchange service, and the real cost considerations. I describe the actual migration process, including the necessary planning and testing. And then there are the gotchas: there are several potential issues that need to be dealt with in such a migration, some of which are not obvious when testing the environment.

Hosted Exchange Services vs. In-House Exchange Solutions

Most businesses start small enough in the beginning that having an in-house Exchange server doesn't make sense. A hosted Exchange service from a reputable provider is a good choice as long as the numbers make sense. When a company gets large enough, the costs for a hosted Exchange service will become expensive enough that it will make sense to examine whether it is time to migrate to an in-house Exchange solution.

The first step in determining when to migrate is to look at the needs of the business and determine if the hosted service meets the growing company needs. One need may concern cost, but other needs may involve features, availability, SLA, security, and SOX compliance, among other business needs. If the need for features and affordability is met by the hosted provider, then there is no reason to change. Some cost considerations determine when to migrate: hardware and software costs are depreciable costs spread out over the life of the installation of the application and are not monthly or yearly costs. Each copy of Microsoft Office includes a CAL (Client Access License) for connecting to Exchange, so it is not necessary to buy Exchange CALs for every user. Exchange CALs are necessary for all concurrent OWA (Outlook Web Access) sessions, and that is a good gauge of the number of Exchange CALs needed. Prices for a hosted solution go up for each new mailbox and with each additional service and are paid every month. At the end of the host email contract, you don't own anything from a hosted service but your data.

There were several reasons that led the company to change to running our own Exchange server. As we grew, it became more apparent that we needed more control and the latest features that only Exchange 2007 would offer. Our hosted provider wasn't planning to upgrade to Exchange 2007 in the near future, and the cost for hosted service was growing far more rapidly than if we ran it ourselves, so we made the plan to migrate.

One of the other advantages of migrating to Exchange 2007 was support for our Mac users. We had about 10% of our users running OS X using Entourage 2008. The connectivity for Entourage 2008 and Exchange 2007 made everyone happy, even me as an Exchange admin.

While Exchange 2007 has many great features for the users, it has added a great deal of complexity. The Exchange 2007 server roles help to define functionality and reduce the attack surface by limiting the roles a server has to provide services for, but this mostly applies in a larger organization. For a rapidly growing company it can be a hindrance. Exchange 2007 resembles Exchange 5.5 in the level of administrative complexity, which is bad, but SP1 for Exchange 2007 helped with some of that. It took a while to work all that out and get the configuration that had both the features and the security we wanted. This delayed rollout of the project by more than a month.

The Migration

Once we had a configuration that worked, we had to develop a plan for migrating all the mail for our users who had their mail hosted on a server we didn't own. This presented several problems. First, we didn't have console access to the hosted mail servers, and we couldn't connect to them in any way that would allow us to transfer data from one server to another. Because of the poorly designed Web interface at the hosted mail service, I couldn't get a list of users, contacts, distribution lists, and members saved to a file. All that had to be recreated by hand. I was eventually able to capture usernames in a messy fashion, save them to a text file, remove all the HTML header info, and then import them into a file I could use for reading in a script. Still not the automated solution I was hoping for, but I don't think the hosted Exchange provider was that interested in providing tools for migrating away from their service.

To avoid downtime, we prepared alternate forwarding addresses for each user from a second email domain, though we could have used a subdomain. The alternate domain was necessary for testing and to avoid email disruption while it was being transferred from the hosting to the in-house service. It may take 24–72 hours for the MX record to be transferred and for that DNS info to completely propagate, so this is probably the best way to make sure email delivery continues until all external DNS information has propagated across the Internet.

Mail forwarding on the hosting service admin console had to be set for each user, but there was no way to do this in bulk at the console. It may be possible to create a script using JavaScript, Greasemonkey, or Perl to scrape the addresses and create the forwarding addresses, but that is likely to abrogate the Terms of Service with the hosted Exchange provider. Please make sure there are no legal or contractual issues before attempting to automate this process on the hosted Exchange provider's Web interface.

In addition, a contact address had to be created for each address that was forwarded to, but that could be automated. Powershell is now the shell scripting language of choice for Exchange; it is simple, it is object oriented, and it works very well in this instance. Something like the following should work to import contacts from a .csv file:

```
$csv = Import-Csv "C:\Contacts.csv"
foreach($line in $csv)
{
New-MailContact -Name $line.DisplayName -ExternalEmailAddress
$line.EmailAddress -OrganizationalUnit "users" -Alias $line.Alias
}
```

Once the contacts are created, the alternate addressing needed to be configured on the new Exchange server. The email address policy in Exchange 2007 made this very easy, as rules could be set to create alternate domain addresses and select a default domain address. Exchange needed to be configured to receive email for both domains. We made sure that a receive connector was in place to accept email from the hosting service and then forwarded a test address to the alternate address. We verified that it was received. Then we verified that the test account could send email and that it showed the correct sending address. We also had that test account send a reply and verified that it was received and showed the correct address. We set the forwarding up for the test account, sent to that account from an outside address, and made sure it was forwarded to the new Exchange server. Then we sent a reply and made sure it was received and showed the correct address.

Once we had recreated the users, conference rooms, distribution lists, contacts, and everything else, we needed to populate them with data. We decided it would be easier to download each user's mail to an Outlook client, saved as a PST file, and then upload to the new server from a new mail profile than to try to get something from the hosting provider, a belief that turned out to be accurate.

Exporting each user's data to a PST file and importing to the new server took a long weekend, but allowed us to migrate the users' email without any downtime for them. As each user was migrated, we sent a message to the user with instructions for accessing their email on the new server via Outlook and via OWA, then forwarded their email from the hosted mail to our new mail server. Then we exported their email and imported to the new server. When all the data was imported, we changed the MX records and let that propagate out to the rest of the world.

Monday morning, the users came in and checked their email and the last message in their mailbox on the hosted server was the message that they had been migrated. Some needed help creating new mail profiles to connect to the new Exchange server, but the process was well documented and tested, and most users had no problems.

Surprises

Now, here comes the bad part: Once we had everyone sending and receiving email from our mail server, we discovered a few problems. First, our conference rooms were not accepting new meeting requests and nobody could see anyone else's availability when creating new meetings. Secondly, and more importantly, replying to messages which were originally generated on the hosted mail solution created errors and the messages were not delivered. In addition, once users had done this, even new messages to the recipient would fail. This is basically the error the sender would see:

```
Delivery has failed to these recipients or distribution lists:
```

```
Username <mailto:%2FO%3DCOMPANY%2FOU%3DEXCHANGE
%20ADMINISTRATIVE%20GROUP%20(FYDIBOHF23SPDLT)%2FCN
%3DRECIPIENTS%2FCN%3Dusername>
```

```
You are not allowed to send this message because you are trying to send on
behalf of another sender without permission to do so. Please verify that you
are sending on behalf of the correct sender, or ask your system administrator
to help you get the required permission.
```

A call to Microsoft Support and 12 business hours later (we didn't wrap up the issues from the call until the next day, or perhaps the day after that) and the Offline Address Book issue and mail delivery problems were fixed. The OAB had to be retargeted, regenerated, and downloaded to most users. Also, the problem with the failed email delivery had to be fixed. The solution, not obvious but not as hard as I had feared it would be, was to create an X500 address for each user which mimicked the old hosted solution server info. That is done by creating a custom address and labeling it as an X500 address. The first part of the address had to mimic the server name of the old server from the hosted Exchange service. That information could be obtained from one of the corrupted email addresses in the OAB. It would look something like this:

```
/O=OEXCH010/OU=FIRST ADMINISTRATIVE GROUP/CN=RECIPIENTS/
CN=USERNAME
```

The first part of the X500 address is the server name of the hosted Exchange service. The second section is the administrative group or storage group the users account was on the hosted service. The third part is the identifier for an email recipient. The last section is the username of the email recipient. This information can be gathered from a test email from a migrated test account or maybe even from the header information of an email, if you know what you are looking for. This info can be gathered from the email header of any email sent from the hosted provider. It would be best to confirm this with the hosted email provider unless you have tested it.

With this and the OAB solution, users could now send and receive email without problems. The final issue was the conference rooms.

The auto-accept issue was simple. That just needed to be set from the Exchange Management Shell for each conference room. The following command will set that:

```
Set-MailboxCalendarSettings <Identity> -AutomateProcessing:AutoAccept
```

It was obvious that the real issue was a rights or permissions issue. I tried granting the distribution list, which included the entire staff, rights to the conference rooms, but that did not work. The correct solution was to create a security group, add the conference rooms to the security group, and grant the distribution list rights to the security group. This solved the problem with visibility of calendar info for the conference rooms as well as availability info for users when scheduling new meetings.

Conclusion

While there were significant issues to overcome, and some unexpected surprises, the migration was cost-effective and resulted in a stable email platform that provided all the features the company needed. I could have benefitted from more testing, particularly testing a migration of two or more mailboxes and a conference room and then testing send and receive between them and outside mailboxes, which would have found the major gotchas that the users discovered. I learned some important lessons about the strengths and weaknesses of hosted services and the types of things that can go wrong when I had only half the info to plan for the migration. I have discovered that for the migration process, it is important to think like a sysadmin, but for developing test cases, I needed to think like a user.