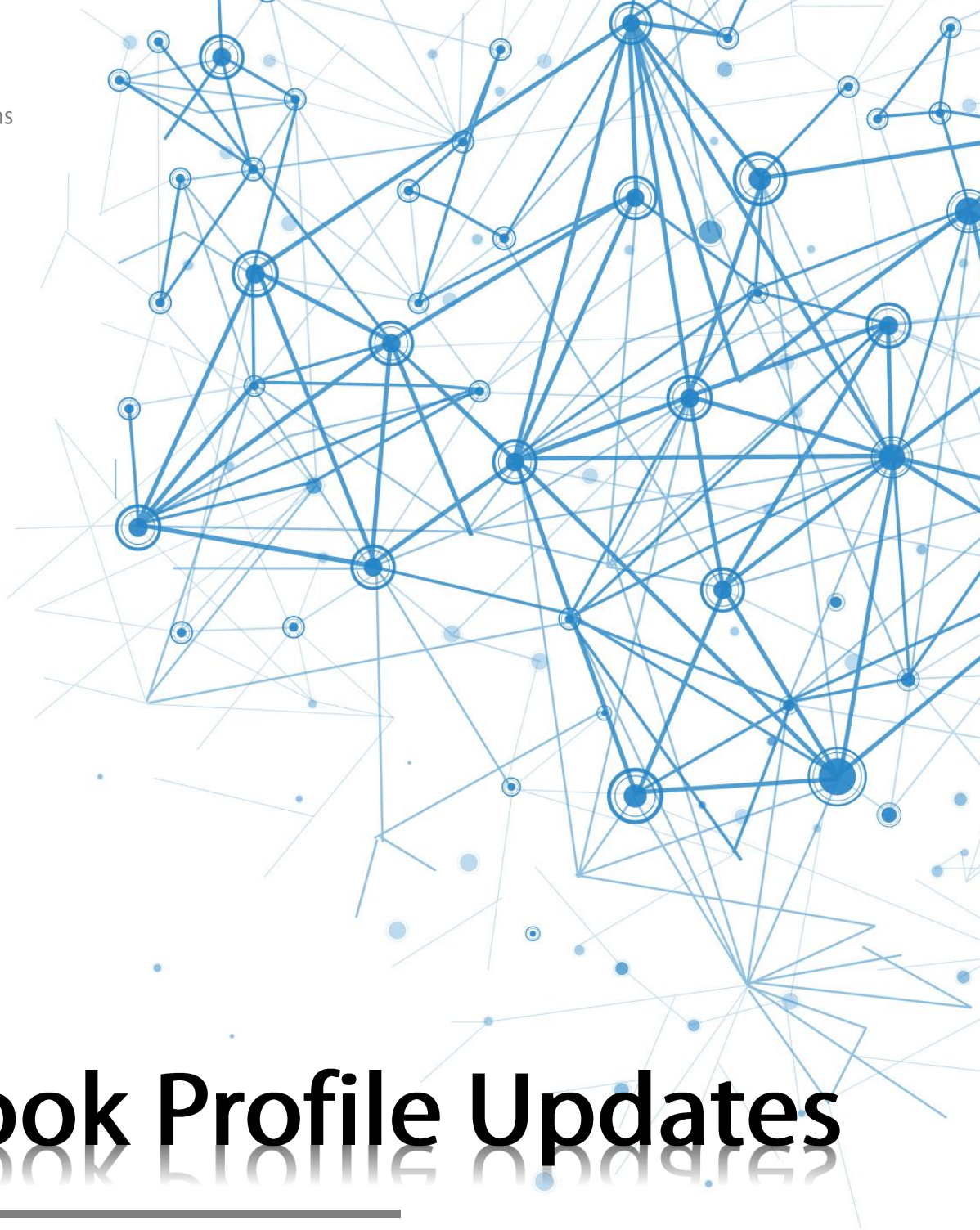
The **Experts**
in Exchange Migrations



WHITEPAPER

Outlook Profile Updates

Advanced concepts for automation and execution of Outlook Profile updates using Priasoft's ProfileManager component

PRIASOFT™
The Experts in Microsoft Exchange Migrations

What is ProfileManager?

Priasoft's ProfileManager is a component of the Priasoft Migration Suite for Exchange. It is a small utility application it is specifically designed to update Outlook profiles as a result of an Exchange migration project. The application is intended to be executed as a user (under their context and credentials) in order to have proper access to the Outlook profile data stored in the registry.

The ProfileManager is a MAPI tool in that is does not directly modify the registry but relies on MAPI to do its work properly with regards to profile modifications. This ensures proper support and use of the Outlook profile.

Document Audience

This document has been written for migration architects, administrators, and team members responsible for the Outlook client and its use. Help desk managers may also find value in this document.

ProfileManager

ProfileManager is an application (~600kb) that is used to modify **existing** Outlook profiles; the application does not create new profiles. The application uses 2 additional files to facilitate the work it performs: "prof.config" and "mbm.puf".

PROF.CONFIG

This file is a simple, INI styled text file that contains key=value settings that control how ProfileManager executes. Items such as Outlook Anywhere and cached mode are controlled by the settings in the Prof.Config.

MBM.PUF

This file is a data file that contains one line of text for each migrated mailbox. Each line consists of several fields of information separated by vertical bars (|). There is purposely sufficient information in this file so that ProfileManager does not require connectivity to any Exchange servers in order to properly update an Outlook Profile.

Execution

ProfileManager, upon starting, looks in the same file system folder in which it resides for the PROF.CONFIG file. If found, it reads the file into memory and sets several internal switches and variables necessary for use. Failure to find or open PROF.CONFIG causes ProfileManager to exit immediately.

One of the key settings in the PROF.CONFIG is the name, and possibly the location, of the data file representing the list of migrated mailboxes. The default filename is MBM.PUF, but is not required or enforced to be this name. This document will refer to the file as MBM.PUF throughout for convenience and consistency.

ProfileManager, after reading in the PROF.CONFIG, will then look for the MBM.PUF file. If this data does not exist or cannot be opened, ProfileManager will immediately exit.

NOTE THAT OUTLOOK PROFILES EXIST "PER USER". IF A CLIENT MACHINE SUPPORTS MULTIPLE USERS (E.G. SHIFT WORKERS THAT SHARE A PC), EACH SEPARATE LOGON WILL HAVE DISTINCT OUTLOOK PROFILE DATA

ProfileManager will then look at all the Outlook profiles configured for the currently logged in user. For each profile, a comparison is made between a key attribute of the profile (the X500

value of the mailbox) and the first field in the lines of text in the MBM.PUF. If a match is found, an update to the profile occurs.

This behavior means that once a profile is updated, it should not match again and thus, by design, is safe to have ProfileManager run multiple times; it will only update once. However, each time the ProfileManager executes, it will analyze profiles again. This is especially important for secondary mailboxes that may be configured in a profile. For example, consider a profile for which a shared mailbox is also attached, but the migration strategy is one where the shared mailbox is not migrated at the same time as the primary mailbox.

After migration, ProfileManager will update the primary mailbox setting (because it matches details in the MBM.PUF), but the shared mailbox will be skipped. For the time period until the shared mailbox is migrated, the profile will show the secondary mailbox, but it will not connect to the shared mailbox because it has not yet been migrated.

When, at a later time, the shared mailbox IS migrated, details will then be added to the MBM.PUF and when the ProfileManager executes again, it will then find a match between the secondary mailbox and the MBM.PUF, causing an update to the profile for the secondary mailbox.

Requirements

The ProfileManager component is, by design, a simple application with regards to deployment and use. There are some requirements to understand:

Must run as the user

ProfileManager must run as the current user.
Outlook profile data is stored in HKEY_CURRENT_USER in the registry and as such applications that need to access such information must run as the currently logged in user. If ProfileManager were to run as an "admin" account, the "admin's" Windows profile would be loaded and likely, on the user's PC, would not have any Outlook profile information to update.

Must run when Outlook is not

Due to the way Microsoft created MAPI and profiles, only ONE APPLICATION can open an Outlook profile for read+write access at a time. As such, ProfileManager cannot update a profile if Outlook is currently running.

Outlook is not the only application that could "lock" a profile. There could be other applications – Lync for example – that lock a profile, even briefly, when they are running or when they first start up. It is for these reasons we recommend that ProfileManager execute during logon, before other applications have a chance to interfere.

Group Policies must not interfere

Group Policy administrative templates exist for Outlook that allow administrators to control many aspects and features of the Outlook client. It is not uncommon in an enterprise to have a template that is used to force a setting for Cached Mode, or even Outlook Anywhere settings.

While convenient for administration of daily life in an organization, such settings impeded the work of ProfileManager. While such a template is employed, an end-user is locked out of any ability to make changes to those features that are enforced by policy. This in turn can affect profile updates since ProfileManager must run as the user. GPOs of this type should be stopped before an Exchange migration is to occur. The settings enforced by the policy do not have to be reversed, only the enforcement of them removed; this will allow ProfileManager to make changes to those settings.

Logs needs full access

The default location for log files produced by ProfileManager is in a folder called "Logs" in the same location as the exe itself. Since this location is most likely a network path (\\server\share\profilemanager, for example), users will need full access to the folder at that location. If a user does not have sufficient permission to create or modify files in that location, ProfileManager will still create a log file, but it will be stored on the user's PC at the following folder

location: %userprofile%\Profile.Logs. This location is inconvenient for tracking the results of possibly hundreds or thousands of users (depending upon your organization size); centralized logging at a network file system is often better.

MBM.PUF must be readable

Much like users being able to read and write to the "Logs" folder, users must also be able to open and read the contents of the MBM.PUF file. A subtle consideration here is related to the account under which ProfileManager executes: if that user does not have permissions to read the MBM.PUF file, then ProfileManager will exit immediately.

Profiles must update before Outlook is used

In modern Exchange deployments the AutoDiscover feature is prevalent and required but also creates issues for migrations. If Outlook is run after a mailbox is migrated, but before (or without) ProfileManager has a chance to update the profile, the profile will NOT BE PROPERLY UPDATED.

The reason is that Outlook does not have code built in to understand that a migration occurred. It consumes details from AutoDiscover only to understand *where* to connect. As such, a pre-existing profile will have many other settings, with values that point at source Exchange servers, Domain Controllers, and Global Catalogs. These additional values are set when the profile is first created, but are not reviewed or modified just because of new or different values returned by AutoDiscover.

Priasoft's ProfileManager has specific intelligence to reset these additional attributes such that the next use of Outlook will cause those attributes to be filled in again (as if it was a new profile), but with details from the new, target environment.

Outlook + AutoDiscover is potentially a frustrating combination for a migration project. If Outlook does run, after a mailbox has been migrated, and ProfileManager was not able to update the profile, the profile is in an unfixable state. The only option from that point is to delete the profile and create a new one.

Deployment Options

Given the small set of requirement, the most critical being that ProfileManager runs as the user, the options for deployment and automation are nearly infinite.

There are a few common strategies we see most customers use for automating the execution of ProfileManager:

Logon Scripts

This is likely the #1 most common strategy for automation and is our recommended approach. Using a logon script ensures that ProfileManager executes before Outlook has a chance to run and also before any other "user" application could possibly interfere. Given the simplicity of execution, a logon script simply needs to run ProfileManager, meaning that there are no command line parameters to be concerned with or other complicated techniques to setup.

There are a few cautionary notes with logon scripts to consider, however.

PROFILEMANAGER SHOULD RUN AT THE TOP

Having ProfileManager run as the VERY first item in a logon script ensures that none of the actions of the logon script can interfere (like changes to the working directory or permissions changes). It also is easier to validate and troubleshoot issues since, as the first item, any logging produced will be also first.

PROFILEMANAGER SHOULD BE IN A DOMAIN POLICY

While a true "group", or even container policy can be used to trigger the logon script, it is best to set ProfileManager to execute for all users in the domain. With such a configuration, there is little chance that a user, or group of users is missed or forgotten. Given the headache that can occur if Outlook + AutoDiscover occurs before an update, it is important to ensure that ProfileManager has a chance to run for all users. Furthermore, given the design of the application, it will NOT update a profile unless that mailbox's data is in MBM.PUF.

Desktop Automation Software

Products like Microsoft's SCCM, ScriptLogic, and Altiris are all good candidates for automating the update of Outlook profiles. The fact that these solutions have an active agent on users' desktops provides an opportunity to interact with an Outlook profile in a more timely fashion than with a logon script.

Priasoft Mailbox Migration Manager, for example, has the ability to call a script after a mailbox completes its migration. Some previous customers have used that event to communicate with the Desktop Automation service to trigger a profile update right after the mailbox completed. Compared to the Logon Script idea where it could be hours or days before the next logon occurs, Desktop Automation could occur near immediately.

In addition, most Desktop Automation solutions provide much more advanced logic for determining the state of a user than a logon script. As such, better intelligence can be used to make a determination of when/if a profile update is necessary (even though this is also handled by ProfileManager).

Manual Execution

While maybe not the best option for a large environment, there are times where manual execution of ProfileManager are necessary. By design, the simple requirement of "must run as the user" helps in these scenarios.

As long as a user can be instructed to run the ProfileManager from the configured location, and if his/her mailbox has been migrated, the profile update can occur. If ProfileManager is deployed to a network file system (as suggested), a user need only be told to run the file directly from that location, for example: \\server\share\profileUpdates\ProfileManager.exe.

This simple path is all that is required. A user could simply enter such in a "Start->Run" dialog, or even in the address textbox of Windows Explorer.

This can also be used as a backup plan for the helpdesk for cases where users report issues about profiles, the helpdesk can respond by first having a user browse to the location and double-clicking ProfileManager.exe.

ZIP File Packaging

This last option is often used for groups of users that fall outside of other options. Common use cases are for users that travel, work from home, or are otherwise not participating in any automation or do not have access to an internal network file share.

Following the completion of a migration batch, one would store ProfileManager.exe, PROF.CONFIG, and MBM.PUF in a zip file (or even a self-extracting zip file) and place this file on a company's internal website, or even on its public site if properly protected (e.g. logon process to access). Such users would be instructed to look for this zip file on the website and would download and run the enclosed application at the appropriate time.

In this way, even the fringes of IT supported users can be addressed since users would be able to self-service their needs following a migration.

An unfortunate side effect

Something that many people don't know is that once an Outlook session is started, there is no way to drop such a session (in a timely manner) for a single user from the server side of the connection.

Common ideas are often tried after reading this statement:

- What if the user is disabled? Nope, the connection remains.
- What if the password is changed? Nope, the connection remains (at least too long to be helpful).
- What if the user account is moved to a different OU, with different permissions? Nope, the connection remains
- What if the user account is deleted? Nope, the connection remains.
- What if the mailbox is deleted? Nope, the connection remains.

Given the strength an Outlook session has, and the fact that in order to perform a proper update of the profile, Outlook must not be running when ProfileManager runs, there can be a challenge here with automating profile updates.

Consider this very common use case: A user is instructed to LOGOFF his/her PC on Friday before going home. The idea being that logon scripts will be used to update profiles. However, users being habitual, many will simply "lock" their PCs instead of actually logging off.

Monday morning, after a migration, a user simply UNLOCKS his/her PC and finds Outlook still running. In a few minutes that user realizes that he/she is not receiving new mail and closes Outlook, followed by re-opening Outlook. At that point, one of 2 things happen: either Outlook fails to open (because the source mailbox has been blocked by migration) or AutoDiscover jumps in the way and attempts to reconfigure parts of the profile.

In either case, the user is in a bad state, and one of the states is not recoverable except to create a new profile (AutoDiscover).

Desktop Automation has a better chance of preventing this case, but such is faced with potentially closing Outlook before a user is ready, or the closing might be interrupted by the user (a cancel dialog, for example).

Advanced Ideas

With the way that Outlook connections work and the inability to interactively drop a connection, except at the user's desktop, some advanced ideas are necessary to encapsulate this issue and provide a better chance of covering all the possible cases.

Forced Logoffs

Using a Group Policy, an admin could create a new group policy that sets specific logon times for users. If the days of the week and times were set such that user logons were not allowed during the migration event, an assurance could be made to see that all users are logged off just as the migration started. In order for this idea to work properly, the policy would need to be enabled for users a day or more ahead of time, but less than a week. This is necessary since policy updates are not propagated to users in real time, but are on an interval that is every few hours to 24 hours (depending upon setup). In order to ensure that all users received the policy change before migration starts, the policy would need to be enabled for users far enough in advance to allow for the update interval to occur.

Once the migration completes, the users could be removed from the policy and return to their normal logon hours.

Forcing a logoff will ensure that Outlook is closed and that logon scripts will execute and will help avoid the unfortunate side effect.

Information Store Restarts

Considering that all users that are connected to a source mailbox database are connecting thru a service running on a server, namely the Microsoft Exchange Information Store service, if this service were to be restarted, such would cause all connected Outlook session to be dropped.

This is not always the best or even acceptable option due that the fact that ALL connections to the mail server would be dropped. However, if properly controlled, or if the environment is small enough such that all mailboxes migrate in a single event, this can be a very good option and can be better than dealing with the "unfortunate side effect".

One idea that can help justify this option is the fact that if all the mailboxes are migrated in a single event, the source mail system would be a dormant system at that point such that stopping the Information Store would have no negative effect. Even while the Store service is down, mail flow would still work since the transport services would still be online. The detractor to this idea is for cases where not all mailboxes (from that server) can be migrated in a single event.

Firewalls

There is a potential for intelligent firewalls to cause Outlook sessions to drop, but such would need to have knowledge of specific user connections and a process to inform the firewall of the state of a user (migrated or not migrated).

This option may not work however if internal users are not connecting thru this device/service.

Desktop Automation

The use of Desktop Automation solutions, as mentioned previously, can potentially manage the issue with Outlook sessions. If a process can be created to close Outlook, run ProfileManager, then restart Outlook, Desktop Automation may be the best option.

The downside of this option are cases of closing Outlook right while a user is in the middle of some action, potentially cause the user to lose changes made. If a prompt is provided, a user may cancel the update in which case it can be difficult to cause a retry without leaving some application running with a "timeout" to cause it.

Shortcut Modification

Since one of the issues with Outlook is AutoDiscover values being applied before ProfileManager runs, another option is to modify the Outlook shortcut(s) on users' desktops to cause ProfileManager to always run first. The idea works like this:

1. A script is created to modify shortcuts pointing to Outlook.exe.
2. The shortcut is changed to run ProfileManager.exe instead of Outlook.exe.
3. The shortcut is modified to retain the Outlook icon and display name (it still looks like the Outlook shortcut).
4. ProfileManager's PROF.CONFIG file is setup to run a program after a profile is analyzed; in this case it would be set to run Outlook.exe.

In this setup, the end user really wouldn't be aware of the change since the icon and name of the shortcut remains the same, only the target of the shortcut changes.

ProfileManager would then be responsible for starting Outlook after it finished. ProfileManager's action to run a program would trigger and run regardless of whether a mailbox had been migrated, ensuring that Outlook would start no matter the case.

The value in this option is that ProfileManager is forced to run before Outlook. It eliminates any timing issues or failures of an event to occur.

The challenge in this option is the initial deployment. Specialized scripts would need to be created that can properly modify a windows shortcut (they are not simple text files). In addition, nearly as much planning and effort would happen to automate this approach as would just running ProfileManager.

Lastly, would be the requirement to clean up later by restoring the shortcuts to their original state sometime after the migration completed.

Conclusions

Proper updates to Outlook profiles has the greatest influence on the perception of success of a migration. If users struggle to gain access to their migrated mailbox, it is very disruptive and generates many helpdesk tickets.

Identification of the likelihood and which mailboxes/users would fall into the various categories is a key part of controlling the success of a migration. It may not be possible to apply technology to all users in an organization in order to address Outlook profiles and as such, alternate plans should be explored, even if it is to delete and create a new profile for key people or VIPs.

If additional question arise from these ideas, feel free to reach out to our support team at support@priasoft.com. We may be able to work on some combination of ideas to help address a concern.