BoxTone®

Exchange ActiveSync Management Challenges

By Paul Robichaux

Executive Summary

Exchange ActiveSync has become the de facto standard for synchronizing mobile device calendar, contact, and e-mail data. A successful mobile device deployment depends on giving users a reliable, stable synchronization experience, but there are many parts in the Exchange ActiveSync data path, and if any of these parts fail or degrades the overall experience for some or all of your users will be substandard. This paper describes the Exchange ActiveSync protocol, the data path used for provisioning and synchronizing devices, describes the major challenges inherent in providing reliable sync services particularly for bring-your-own-device (BYOD) users, and discusses how BoxTone's solutions help meet those challenges by improving uptime and availability, reducing mean time to repair (MTTR) and increasing first-call problem resolution rates.

About the Author

Paul Robichaux is a Microsoft Most Valuable Professional (MVP), Exchange architect, writer, and software developer who has been working with Exchange since 1996. He is a senior contributing editor for *Windows IT Pro* magazine and the author of *Exchange 2013 Inside Out: Clients, Connectivity, and Unified Messaging* (Microsoft Press; October 2013).

Introduction

Smart mobile devices, once the stuff of science fiction, have become incredibly commonplace in the industrialized world. Companies of all sizes are finding that their users are demanding to pick their own devices and then connect them to company-provided resources; the gains in employee productivity and flexibility make this a worthwhile trade in most cases, but mobility opens up a host of potential problems and allowing full-blown "bring your own device" (BYOD) deployments only compounds these. In this paper, we'll describe how Exchange ActiveSync works to enable users to provision and synchronize their own devices, highlight the key challenges that companies relying on Exchange ActiveSync (EAS) face, and discuss some ways to reduce the risks that may come from allowing mobile device, particularly BYOD devices, on your messaging service.

The paper will also include both discussion of the built-in monitoring and logging capabilities of Exchange, as well as third-party solutions that address problems Exchange's built-in capabilities miss. The paper will particularly look at BoxTone, a Mobile Service Management solution that provides real-time monitoring, diagnostics, and alerting for Exchange ActiveSync. It will cover not only how Exchange admins can use traditional tools, but also how both administrators and mobile operations and support staff can utilize new emerging solutions for Exchange ActiveSync management to better support and ensure reliability of their messaging environment.

Exchange ActiveSync Explained

The name "Exchange ActiveSync" actually refers to three different, but related, things, and separating them clearly is a good first step to understanding how EAS works. EAS is

- a family of protocol specifications published by Microsoft. These specifications set out the operations that EAS clients and servers may, may not, and must perform, how clients and servers exchange data, and so on. For example, the protocol specifications describe how an e-mail message and its attachments are presented to the device, how the device should retrieve them, and how the device can mark the message as read or unread As with most other similar protocol specifications, they aren't exciting reading, but because they clearly define exactly what clients and servers are supposed to do, they're important to developers.
- a server-side implementation; for example, Exchange Server has included EAS since Exchange 2003 SP1, and even some competing products from companies such as IBM Lotus and Kerio include EAS server support so they can talk to the clients mentioned below
- a device-side implementation, or *client*, that talks to the server. Some mobile device platforms, such as those based on Apple iOS, include EAS client

support as part of their base product, while others (such as Google Android) may or may not include onboard EAS support depending on the device manufacturer. In this paper we will use the terms "device" and "client" interchangeably to refer to any EAS client no matter what platform it's running on.

EAS support

When we talk about an EAS feature or behavior, it's important to remember that all three of these pieces are involved: the specification defines what's supposed to happen, but the device and server implementations must correctly implement that specification in order to produce the desired results. There's not much enforcement involved in the protocol; for example, Microsoft doesn't check device implementations to verify that they conform to the protocol specification, and the server generally doesn't (and, in most cases, can't) verify whether the client is telling the truth about applying EAS policies or settings.

Furthermore, clients implementing Exchange ActiveSync vary widely in terms of which aspects of the spec they implement. While there is a base set of features all clients must implement, this list is fairly rudimentary. For example, a client may choose not to implement calendar or task sync, or email flagging, or other features at their option. In a few instances, clients have chosen to use Exchange ActiveSync only for syncing contacts, ignoring email entirely. The growing list of device OSes, versions of these OSes, OEMs (particularly for Android) and carriers has led to a proliferation of clients, no two of which implement EAS in exactly the same way. This substantially increases the chance of issues, ranging from relatively minor issues impacting the user to substantial issues that can impact an entire Exchange environment.

Microsoft maintains a page showing known EAS issues with different clients at http://support.microsoft.com/kb/2563324 but it's not guaranteed to be a complete list. It only reflects issues that Microsoft knows about and is tracking, so as new issues emerge, they won't be on the list until Microsoft verifies them and has started working with the EAS client implementer. When new bugs emerge (for example, when a device manufacturer a new revision of its mobile OS that contains new EAS client bugs), it takes a while for the issue to be reported to Microsoft, analyzed, reproduced, and added to this list. To cite one example of the way this process works, in the spring of 2013 Apple shipped version 6.1 of iOS, and it contained a bug that could overwhelm Exchange servers with an excessive volume of transaction logs. Administrators began reporting the bug, both formally to Apple and Microsoft and informally through social media and support forums. However, Microsoft didn't officially add that bug to the list until it had communicated with Apple, verified that the bug existed, and determined that the bug was due to changes Apple had made, and the bug itself couldn't be fixed without a software update on each affected device. Many organizations with iOS clients first became aware of the bug when their servers suddenly started performing poorly; they had no proactive means of receiving notification or of determining the root cause of the problem.

The EAS data path

It's also very important to remember that the overall EAS experience depends on more than just the three components described above. For example, a well-behaved EAS client and a properly configured server are still subject to network interruptions, mobile carrier connectivity problems, and so on. Figure 1 shows the entire path from the device to the user's mailbox; any problem with, or interruption in, any component in the path will result in a poor user experience, excessive load on the server, or possibly both.

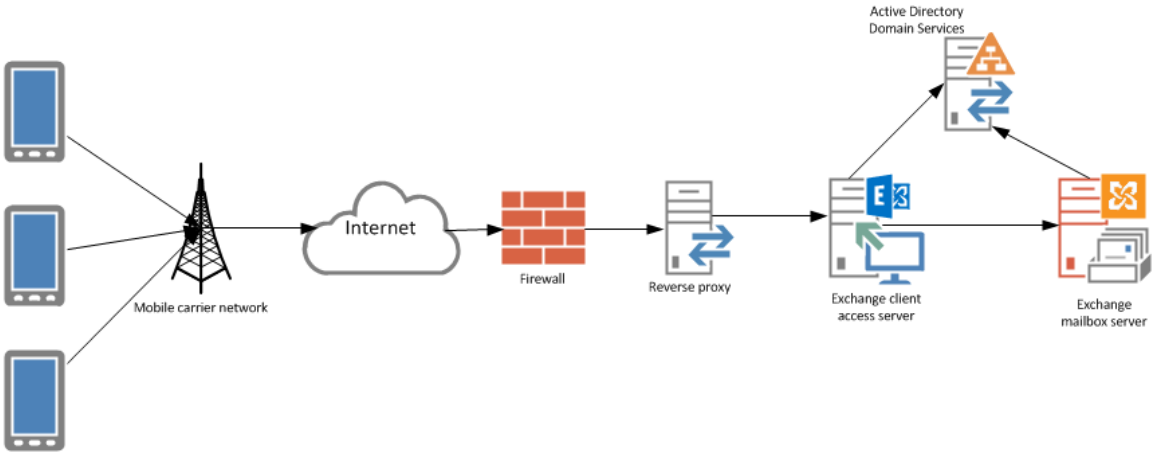

Figure 1: the EAS data path

The complex nature of this data path means that there are multiple points of failure that can cause problems for one, some, or all of your users:

- Hardware or software failures on an individual client may render it unable to connect to the network or sync.
- A bug in a new release of a client OS may overload the CAS or mailbox servers or render clients running that OS release unable to sync properly; this is essentially what happened with the Apple iOS 6.1 sync bug.
- A failure on the mobile operator's network may keep devices from being able to connect and sync; these problems can be particularly hard to track because as devices move around between cellular and Wi-Fi coverage they may suddenly start, or stop, working.
- A failure of the firewall, reverse proxy, or load balancer can prevent devices outside the firewall from synching without affecting those inside the firewall.
- In an Exchange topology that contains multiple CAS and mailbox servers, failure of any one of these servers will affect users who sync with those servers while possibly leaving other users completely unaffected.

BoxTone's unique approach to monitoring all parts of the EAS data path means that it can help you identify the source of a problem—whether it's in the device, mobile operator network, or in a part of the infrastructure that you control. This helps solve one of the major challenges in managing Exchange ActiveSync: dealing with these

problems quickly and efficiently—including finding out that a problem is happening before users start to complain!

How EAS devices synchronize

Exchange ActiveSync uses a technique known as a *hanging sync* to transfer information. This type of sync takes advantage of the fact that HTTP connections may be left open indefinitely; neither the client nor the server actually has to send any data over the connection to keep it open. Mobile devices maintain two separate radio connections to their mobile network: one connection transfers data, while the other is reserved for control commands. By putting the data channel into an idle state and the control channel into a low-power listen-only mode, a mobile device can essentially idle its radio while waiting for the server to notify it that new sync data is available, so keeping a hanging sync open doesn't have a huge effect on battery life. (EAS includes provision for dynamically adjusting the length of the hanging connection to accommodate network timeouts imposed by device manufacturers or mobile carriers). The advantage to using hanging synchronization is that it allows the server to send data to the client as soon as it arrives, providing push synchronization and rapid mail delivery.

The actual synchronization process starts when a client contacts an Exchange Client Access Server (CAS) to request synchronization. The synchronization process requires the device to reach TCP port 443 on the Exchange CAS so it can make HTTPS requests to the Microsoft-Server-ActiveSync virtual directory. Any problem that prevents the device from contacting this particular virtual directory on the target server will impair synchronization. This might include problems with the internal or carrier networks, permission problems with the CAS server, failure of the Exchange Autodiscover process, or a problem with the CAS role itself.

The client's initial request doesn't have anything to do with synchronization—first it has to know which version of EAS the server speaks. Typical mobile device clients can work with version 12.0 (the Exchange 2007 version) or later of EAS, but there are some implementation changes from 12.x to 14.x, so the device needs to know how to interpret server responses. The client can discover the server version in one of two ways. It begins by sending the HTTP OPTIONS command to the server; if the client gets no response (usually because a firewall or reverse proxy blocks that particular HTTP command), or a response that doesn't include the server's version data, it instead asks for an actual synchronization with a special flag value indicating synchronization request with a synchronization key of 0.

In either case, the server responds with two headers indicating which EAS version and command set it supports. Once the client sees those headers, it can request an actual synchronization. Part of the request the client sends is a policy key that indicates what current EAS policy the client has applied. If the client doesn't have a policy, or has an outdated policy, it needs to be provisioned, as described in the "Device provisioning" section later in the paper.

Assuming the client has a current policy and isn't blocked or quarantined by the server, it can issue one of three commands: `FolderSync` (for synchronizing the hierarchy of available folders), `Ping` (to indicate that the client is interested in knowing if there are new items available), or `Sync` (indicating that the client wants to sync items in a specific folder). Each of these must include a sync key that tells the server which specific folder the client's requesting. One sync key is used for folder hierarchy synchronization, and then each individual folder that the user or device selects for sync has a unique sync key. That way, the client can sync only the folders it wants instead of all folders.

Both the `FolderSync` and `Sync` commands basically ask the server to send any new items that arrived or changed since the previous successful sync. The `Ping` command implements a hanging sync by asking the server to send a response only when new data become available, at which point the client sends a `Sync` or `FolderSync` command. Here's how the hanging sync process works:

1. The device issues a request, which has a default lifetime of 15 minutes (the interval is known as the heartbeat interval). The request signals the server that the client wants notification of any item that changes in any folder that's tagged for synchronization. The list of folders to synchronize is established during the initial device synchronization and updated any time the device user changes the folder list.
2. Until the heartbeat interval passes, the client doesn't expect an answer unless new items appear.
    a. If the server doesn't send back any response, the device can idle its radios and wait for a command signal indicating incoming data over the command channel from the mobile operator to wake them up again.
    b. If the network connection times out, the client makes a new request with a shorter heartbeat interval.
    c. If the heartbeat interval passes and nothing has changed, the server sends back an empty HTTP 200 OK response. The client then goes back to step 1.
3. If the client receives a 200 OK response with data, the contents of the data it receives will indicate what changed on the server. The client should apply these changes to its local data store.

What about new items generated on the client? As soon as an item changes locally (perhaps you've sent a mail message, accepted a meeting invitation, or created a contact), if the client is online it will immediately initiate a sync. The EAS protocol includes several commands just for this situation; for example, the `SmartReply` command is for clients to use to send a reply to a message without uploading the entire text of the original, and `MeetingResponse` is for accepting or rejecting meeting invitations. Not every EAS client uses these commands; for example, iOS versions before 5.0 didn't. If the client isn't online, it should batch local changes and

send them to the server when it reconnects; some clients give users a better offline experience than others.

Device Management in Exchange 2010 and Exchange 2013

There are basically four parts of the device management lifecycle in Exchange 2010 and Exchange 2013[1]. (Although Exchange 2007 EAS has many similarities to the process described here, there are some critical differences: it supports fewer EAS policy settings, and the server lacks the allow/block/quarantine feature and has much more limited logging and reporting than its successors.) Devices must be provisioned with policies; the server must decide whether or not to allow a particular device to connect; the server must be able to command the device to erase itself remotely, and the server and its administrators must be able to monitor and log these operations, along with normal synchronization and any errors or exceptions that may occur.

Device provisioning

The notion of BYOD is that an individual user can unbox a new device, personalize it however she wants, and then connect it to her organization's network through EAS, all without any help (or interference) from the organization's IT staff. In order for this to happen, EAS has to have a way to push policies to a device, to verify whether a device claims to have applied a policy or not, and to tell the client to re-apply the policy when it changes. (Note that EAS does *not* include a way to tell whether a device is telling the truth about its policy state!) Device provisioning is the mechanism that EAS uses to accomplish these tasks.

The client includes a policy key with each request it makes; the server examines the policy key supplied by the client, and if the client's policy is older than the current applicable policy, the server returns a status code that tells the client to request an updated policy. When the client requests a policy, it does so in a two-step process that results in it signaling the server whether the client applied all, some, or none of the policy. This gives EAS a crude way to make decisions about whether a device should be allowed to synchronize or not. Notice that the client doesn't have any way to choose a specific policy; the server is in charge of deciding which policy to assign based on the assigned EAS mailbox policy for the user's mailbox.

When a new device is first provisioned for a user, that device is said to be paired with the mailbox. By default, each user may have up to 10 paired EAS devices (that limit is configurable). One common user complaint is that one or more devices won't sync properly while others work fine—this can be tricky to identify and fix.

[1] We can treat these two versions of Exchange as identical from an Exchange ActiveSync perspective; there are no major changes in the EAS protocol or implementation between the two, and where there are minor relevant differences they'll be highlighted.

The EAS provisioning mechanism is fairly limited compared to third-party mobile device management solutions, which is why a large number of companies have turned to them.

The allow/block/quarantine mechanism

Exchange includes a feature that lets you define *device access rules* that determine whether a particular device type or platform is allowed to synchronize or not. You can define rules for individual devices or platforms that either allow or block those devices; you can also set a default behavior that specifies what happens with devices that don't have a specific rule. As an additional layer of coverage, you can allow or block sync for a particular user/device combination. The process of applying these rules is known as allow/block/quarantine, or just ABQ, and it follows nine steps when a client connects:

1. Is the client able to authenticate? If not, it obviously can't sync so no further action is required.
2. Is the user who just authenticated enabled for ActiveSync? If not, again, no further action is necessary.
3. Does the client claim to have applied the correct EAS policy for the EAS mailbox policy set for the user? If the answer is "no", the connection may still be allowed if the policy allows non-provisionable devices to connect.
4. Does the user have a personal exemption that allows this specific client?
5. Does the user have a personal exemption that blocks this specific client?
6. Is the client blocked by a matching device access rule?
7. Is the client quarantined by a matching device access rule?
8. Is the client allowed by a matching device access rule?
9. If the client has made it this far, the server will apply the default access level (allow/block/quarantine) specified in the ActiveSync organization settings.

EAS remote device wipes

The ability to remotely erase a device after it's lost or stolen is a key feature in EAS. When an administrator or user tells Exchange to wipe a device, the server sends an EAS wipe command to the device, which is supposed to carry it out and then acknowledge to the server whether or not the wipe succeeded.

Exactly what is erased may vary between devices; the EAS standard says that the client should destroy all data and credentials that were ever used with the server. Some clients (such as Apple iOS and Microsoft Windows Phone) erase everything, but others might not erase removable storage cards, or they might erase only data that came from the EAS account that requested the wipe. This difference in behavior is a strong argument in favor of testing device features and using appropriate allow/block/quarantine rules (as described later) before approving a device type for deployment. Exchange sends a confirmation email when a device acknowledges a wipe request; if a user requested the wipe, he receives the confirmation message, and if an administrator did it, the administrator and the user both get messages.

It is critical to understand that as currently implemented, this feature is a best-effort attempt to make sure the device is erased; there are no guarantees. A crafty thief or attacker may be able to prevent the device from synchronizing, and without synchronization the remote wipe command won't be received and executed. The wipe command remains in effect on the server and will be resent to the device any time it connects. Because the wipe command is best effort and can often be evaded, many organizations find it insufficient for their security needs and rely on third-party Mobile Device Management solutions instead.

EAS monitoring and logging

One of the current weak spots in the Exchange Server EAS implementation is its monitoring and logging infrastructure. Because EAS takes place over HTTPS, every interaction between client and server is logged on the CAS in the IIS logs. However, these logs are maintained separately on each CAS, making it challenging to get a good overall picture of what's happening on your servers. Microsoft has attempted to address this by providing the Export-ActiveSyncLog cmdlet in the Exchange Management Shell; Export-ActiveSyncLog can produce six types of reports:

- The Exchange ActiveSync Usage Report shows counts of the number of items sent and received, organized by type (e-mail messages, calendar items, contacts, and tasks are supported). The reports also show the total number of bytes sent and received.
- The Hits report shows the total number of devices that requested synchronization, as well as the total number of sync requests handled per hour.
- The HTTP Status report summarizes all the different HTTP response codes, including errors, generated by clients.
- The Policy Compliance Report shows how many fully compliant, partially compliant, and non-compliant devices are currently synchronizing with the server. The degree of compliance reported is based on whether the device acknowledged full, partial, or no compliance with the supplied policy when it was provisioned.
- The User Agent report shows the total number of users and which clients they're using, as reported by the client's UserAgent HTTP header. This header normally contains both the client type and the client software or mobile device OS version.
- The Users report lists all the users, and their devices, that synchronized with the server.

These reports are generated as CSV files, with each IIS log generating a separate set of files. If you use the default settings, that means you'll get one IIS log per day. You can run a command such as the following to process these logs:

```
Get-Childitem C:\inetpub\logs\LogFiles\W3SVC1\ |
    where-object {$_.lastwritetime -gt $DateToCompare} |
   ForEach { Export-ActiveSyncLog -FileName $_.FullName
               -OutputPath "C:\reports"
               -OutputPrefix $_.Name.Replace(".log","_") -UseGMT:$true}
```

For a 30-day period, that will give you 180 log files that you must analyze—per server. You can develop more sophisticated scripts to gather this data across multiple servers but that's a non-trivial undertaking. Once you have a set of scripts to process the data, you still have to run them. What's more, this data is not pulled in real-time, meaning the data is often substantially out-of-date before the administrator has an opportunity to interrogate the logs —contrast this with the automated process by which BoxTone analyzes the servers' logs in real time across the entire deployment to give you live data.

Outlook Web App includes an option to collect synchronization data into a report that is stored as an attachment on a new message in the user's inbox; the user can then forward the log, provided their e-mail is working. The log message is generated on the server, not the client. In addition, many EAS clients can produce client-side log files, although the contents and access methods vary from client to client; because the client must enable these logs and then provide them to the administrator, this logging is only useful as a reactive tool for solving an evident problem.

What about Managed Availability?

Exchange 2013 adds a new feature known as Managed Availability that attempts to monitor the health of services, including Exchange ActiveSync. The Managed Availability subsystem uses protocol- and service-aware *probes* to check the performance and availability of individual services and can take actions to try to restore service quality ranging from recycling an IIS application pool to bug-checking a server. The idea behind Managed Availability is that a problem should only be escalated to a human's attention if the Managed Availability subsystem cannot resolve it. This is a promising new feature in large part because it helps make the large-scale Exchange infrastructure required for Office 365 more manageable. However, its design goal is to let servers monitor *themselves*, not the health of the entire data path. Its newness means that most administrators don't have enough operational experience and feedback on it yet to judge its effectiveness. By comparison, BoxTone's monitoring and analytics is based on more than 10 years of experience monitoring performance and uses live data collected from *all* the client access servers in the organization and analyzed in real time to produce proactive notifications and data for reactive action.

What about the outside world?

Exchange logging and monitoring has little or no visibility into what's happening beyond the network boundary. Exchange doesn't have the ability to analyze or correlate log data into actionable patterns such as "all the clients with firmware revision x.y have dropped offline in the last 30 minutes" or "no device on the mobile operator Z network has been heard from in 15 minutes". However, this information

is all in the CAS logs: they contain information about the device itself, how and when it connects, and what the results of those connections are. BoxTone offers a synoptic view that combines all of these factoids to give you a clear warning when a failure affects a non-Exchange part of the EAS data path so that you can isolate and fix it.

Managing EAS Challenges

EAS is a robust and mature protocol, and the Exchange implementation of EAS has been proven by tens of millions of deployed users over a period of many years. When all the components in the EAS data path are working properly, users get a smooth, reliable sync experience—but what about when one or more components *aren't* working right? For example, the failure of a cell tower near one of your facilities may cause a sudden drop in sync activity because devices on that tower are no longer able to reach the network. That in turn may lead to a rash of service desk calls as users discover that their devices are out of sync. A client-side bug included in a device software update may have little effect at first until the number of deployed clients reaches a tipping point, leading to unexpected and hard-to-diagnose problems with synchronization.

Sizing, scalability, and performance

As with every other aspect of Exchange, EAS depends on server performance. Microsoft's comprehensive Exchange server role calculator provides great guidance on how to size mailbox, CAS, and multi-role servers, but, as the documentation points out, there's no substitute for load testing in your own environment; the factors used by the calculator represent Microsoft's experientially-derived guidance, but one size may not fit all. More importantly, Microsoft doesn't provide any tools that tell you when your server's underprovisioned: you can use the built-in Windows performance monitoring tools to watch resource usage, and if you build a baseline of performance data you will be able to see when resource usage or performance falls outside the normal range. However, there are many components in the EAS data path that aren't easily monitored by these tools. For example, a problem with IIS on a CAS, or with the Microsoft-Server-ActiveSync virtual directory, probably won't generate any indications in performance data. There may be items logged in the server's event logs, but then the burden is on you to identify which server is having a problem, which requires multi-server monitoring and analysis itself. If you depend on the built-in tools included with Exchange and Windows, the first warning you are likely to get of EAS problems will come from user complaints because those tools provide information that you can use for *reactive* response, but they don't provide much you can use for proactive problem-solving or troubleshooting.

BoxTone's monitoring architecture collects logs from every CAS and consolidates the log data, using Microsoft's native tools—not installed agents-- to gather log data and statistics, then providing smart analysis, *proactive* alerting, and reporting to

make it easier to see what's going on with any or all of an organization's connected clients. Administrators have access to a dashboard showing key indicators of organizational EAS health, so even when everything is working normally it's easy to keep track of the overall state of the data path as a whole.

Service monitoring, not just server monitoring

Part of the reason that Exchange 2013 Managed Availability has gathered attention is because it marks a shift in Microsoft's emphasis from monitoring *servers* to monitoring *services*. This shift reflects what users see: a particular server can still be powered up, on the network, and running all its services and yet not be able to answer user requests in an acceptable manner. Conventional server monitoring will tell you when the entire server fails, but it's more important to know when the service that server offers is degraded or unavailable.

There are drawbacks to Managed Availability, though. First, it requires Exchange 2013, an obvious problem for organizations that are not yet ready to move to Exchange 2013 or Office 365. In addition, the probes and actions for Managed Availability are preset. This is reasonable, given that the development teams for each Exchange feature and component develop probes and actions that make the most sense for those features and components, but it gives you very limited ability to affect what happens when Managed Availability thinks it has detected a fault. Managed Availability also can't alert you to an impending or potential fault; it doesn't kick in until a probe indicates that something has already gone wrong.

Perhaps more seriously, Managed Availability is an expert tool, built for use by experienced and knowledgeable veteran administrators. Managed Availability data isn't necessarily visible to—or understood by—service desk or network operations center (NOC) staff, who are likely to be the first few tiers of response for reported mobile device issues. Depending on Managed Availability as a tripwire for problem awareness thus means that the Exchange administrative staff has to be continuously monitoring it and pushing reports to the NOC and service desk.

Another major drawback to the use of Exchange's built-in service monitoring tools is that they only "see" service quality inside the Exchange organization. Managed Availability, and its predecessor tools, can see the state of Exchange servers (and the other roles on which they depend, including Active Directory and DNS servers) but they don't currently look at the endpoint, the endpoint's connection to the network, the internal network itself, or firewalls or reverse proxy servers that EAS traffic must traverse to reach Exchange.

BoxTone addresses this problem by gathering data about every component of the EAS data path. By performing in-depth analysis of the CAS logs and surrounding infrastructure, BoxTone knows, in real time, which users are connecting, which mobile devices or clients they're using, which mobile operators are involved, and how the mobile network (if any), firewall or reverse proxy, internal network, and Exchange components are performing. Through active monitoring, BoxTone creates a set of learned baselines of normal behavior for each user, device client, CAS,

carrier and infrastructure component. Abnormal conditions are immediately signaled to the service desk, NOC staff, *and* Exchange administrators so they can identify and fix the problem, whether it's on a single device or an entire organization. The BoxTone alerting system monitors both high and low conditions of the baseline range, so if an unusually large *or* small volume of sync traffic is occurring as a deviation from normal, administrators will be notified and can start looking into it immediately. In addition, because BoxTone allows you to track and correlate failures or swings in activity level to device type and model, mobile operator connectivity, and user, it's much easier for you to identify patterns of failure brought on by device upgrades, systemic problems with connectivity or server configuration, or other root causes that affect multiple users.

Providing user and device support

In a world where everything worked properly all the time, user and device support would be trivial. In the real world, however, Exchange administrators and service desk staff face several tricky problems:

- Because not every piece of the EAS data path is directly visible to administrators, pinpointing the source of the problem can be frustrating and slow. Identifying when multiple users have the same problem may have to wait until multiple users have *reported* the problem, which slows down problem resolution considerably.
- Some problems don't leave obvious signs on the device. For example, a user whose mailbox is over its "prohibit send" quota won't be able to send e-mail from an EAS device, but EAS doesn't return any kind of error code that the client can display to tell the user *why* her e-mail can't be sent, and the service desk won't have that information directly at hand either. So the e-mail just sits there… leading to service calls that could be quickly resolved if the endpoint was able to tell the user exactly what was wrong.
- Not every organization has a way for users to report problems. Some of those which do depend on e-mail for problem reporting, which doesn't work well for users whose e-mail service is affected.
- Service desk and IT staffers often get nebulous problem reports from users: "my tablet stopped getting e-mail" or "I can't see appointments on my phone." These reports aren't actionable by themselves, so they require staffers to dig into the problem and try to isolate its cause directly.
- Many users have multiple devices that sync to the same mailbox. These devices may be synchronizing through different CAS servers, and Exchange doesn't make it easy to figure out what the sync state of each device is.

The result of these problems is that it can be very difficult to identify *which* devices aren't working, *why* they aren't working, and *what* to do about it. The result of this difficulty is striking: based on benchmarking done by BoxTone prior to deployment at a number of customers, the mean time to repair (MTTR) for device sync problems is approximately 30 minutes, and on average 80% of those cases have to be escalated to senior staff in order to reach resolution.

BoxTone helps resolve these problems by giving the service desk an easy to use dashboard that clearly identifies problems all along the EAS device path. Administrators can see all of the organization's devices to make it easy to identify those with problems; this helps identify patterns, too, such as a poorly performing CAS or an outage with a particular mobile device carrier. Because all components of the EAS data path are monitored, pinpointing the root cause with BoxTone is much faster than doing so manually, and this helps drive down the MTTR for sync problems. BoxTone's data indicate that its customers have a MTTR for sync problems of 5 minutes, and an average escalation rate of only 20%: compelling evidence that its approach of live monitoring and analysis makes a big difference to end users.

Conclusion

Users insist on having mobile device access to their critical e-mail, calendar, and contact information. Whether they bring their own devices or you provide them, the management challenges of supporting a pool of diverse mobile devices are the same: spotting potential problems before they blow up; receiving early notification of problems caused by device updates, carrier outages, and other factors outside your control; and quickly identifying the exact source and scope of the problem. BoxTone helps meet these challenges by giving you real-time insight into the health of every part of the EAS data path and showing you clear, actionable data on which devices and servers are healthy and which ones are not—giving you faster MTTR for problems, a lower escalation rate, and a more productive experience for your administrators and mobile device users.

About the Author

Paul Robichaux is a Microsoft Most Valuable Professional (MVP), Exchange architect, writer, and software developer who has been working with Exchange since 1996. He is a senior contributing editor for *Windows IT Pro* magazine and the author of *Exchange 2013 Inside Out: Clients, Connectivity, and Unified Messaging* (Microsoft Press; October 2013).

About BoxTone

BoxTone is the innovator of automated Enterprise Mobility Management (EMM). With millions of mobile devices and apps under management, BoxTone's automated EMM platform is trusted by more of the world's leading enterprise, Managed Service Providers and government agencies than any other—including 41 of the Fortune® 100 and 8 of the Top MSPs—to ensure maximum mobile performance and security at the lowest cost and risk. Only BoxTone's single unified mobile management platform powered by patented real-time automation technology addresses the entire mobile lifecycle: mobile device management (MDM), app management (MAM),

support management and operations management. And only BoxTone delivers real-time, centralized control of all mobile smartphones and tablets including iPhone and iPad, Google Android, BlackBerry and Windows Phone, as well as the enterprise apps that run on them.

BoxTone's unparalleled EMM innovation has also been recognized by leading industry analysts, as the company has recently been positioned in the "Visionaries" Quadrant of Gartner's Magic Quadrant for Mobile Device Management (MDM) Software, named an "innovator" in the Forrester Research, Inc. Market Overview: *On-Premises MDM Solutions* and named to the Winner category in Yankee Group's *MDM is Dead. Long Live EMM!* Learn more at www.boxtone.com, or call +1 410.910.3344.