CHAPTER 1

About Unified Messaging

The essence of communication is breaking down barriers. The telephone, for instance, breaks distance and time barriers so that people can communicate in real time or near–real time when they are not in the same place. But as communications evolve, there are new barriers to be overcome. New forms of communication emerge, such as e-mail, voice mail, fax machines, and pagers, and people use different devices for each of these. Unified messaging (UM) is designed to overcome these barriers. It breaks down the terminal and media barriers so that people using different technologies, different media, and different terminals can still communicate with anyone, anywhere, at any time. Unified messaging is the integration of several different communications media so that users can retrieve and send voice, fax, and e-mail messages from a single interface, whether that is a wireline phone, a wireless phone, a PC, or an Internet-enabled PC.

This chapter introduces the concept of unified messaging and discusses how the technology, along with Unity, has evolved. The topics covered in this chapter are ones that mostly concern organizations, including who manages unified messaging and why, organizational perceptions, and security issues related to converging voice messages with e-mail. Some attention is given to different ways that a user community might adapt unified messaging, as well as what the implementer and manager of unified messaging should expect. Finally, messaging technologies are covered along with legacy switching technologies and Cisco IP Telephony call-processing systems, such as Cisco CallManager.

The Evolution of Unified Messaging

When the concept of unified messaging first was introduced, it appeared to be a grand idea. However, as the technology was implemented, the original idea behind unified messaging was lost. Some vendors developed products that were so purely unified messaging (meaning that they were totally dependent upon the third-party messaging system they were servicing; without this dependency, it couldn't stand on its own) that their very existence was dependent on the messaging system they were installed to service, including the mailbox store and directory. Cisco Unity was one of these pure UM products. Thus, although Unity realized early success, there was a lot to be desired. Reliability and the capability to sustain voice messaging (VM) when the messaging system was unavailable were serious challenges to the early implementations of the product.

Some of Unity's competitors went to the opposite end of the UM spectrum and came out with what the industry calls integrated messaging solutions instead of unified messaging. With integrated messaging, a client deployed at the end user's desktop unifies the messages from the two disparate systems (e-mail and voice). Many companies thought it made sense to deploy these types of solutions. However, if you add fax to this, you mainly have a confused mix of implementations that can vary as much as the features that a given vendor provides in a product. The problem with integrated messaging highlights the fact that, with the current developments in communication, standards are important. Products that offer interoperability are needed. These products might not be from the same vendor, but they must operate together to form powerful solutions for customers.

One thing is certain: The whole idea of unified messaging stirred the messaging industry. Although many options existed, pure e-mail-only systems left less than suitable solutions for companies of any size. The addition of fax vendor support and other multimedia applications integrated into legacy e-mail systems made any notion of useful, easily implemented standards nearly impossible. For the most part, some of these systems have very strong and capable implementations (Microsoft Exchange, Lotus Domino, and Novell Groupwise, to name a few), but this disparity in implemented standards essentially makes it necessary to write code using the proprietary application programming interfaces (APIs) of these messaging vendors. A good example of this is the way in which IMAP is implemented among these applications. Many similarities exist, but there are enough differences to make it challenging to write one solution for all. Thus, for integrated solutions, useful technologies, such as drag-and-drop, would make you think there is more interoperability than there really is.

For unified messaging, the opposite issue is true. Unified messaging solutions such as Unity use the APIs provided by the messaging vendors in lieu of using proprietary ones (to access mail and directory resources). The challenge of using third-party messaging APIs (such as Microsoft's MAPI or the Lotus Notes API) is that, although it is possible to write very functional and capable unified messaging solutions, these messaging APIs lack real voice-messaging support. Instead of treating Unity like a voice-messaging interface into the messaging environment—in other words, like a voice-messaging client—Unity is relegated to acting just like an e-mail client. Unity must compensate for this by accounting for the differences in behavior between an e-mail client and what a voice-mail client should be when accessing these messaging systems.

If the notion of a voice-mail client actually existed within a given messaging system, it would have a different set of characteristics than a legacy e-mail client. These characteristics include giving unified messaging clients—more basically, voice-messaging clients—higher priority when accessing the system than e-mail or other types of clients. This is because the key to voice messaging is the capability to provide a real-time voice interface (such as the Unity Telephone User Interface, or TUI) into the subscriber's messaging store, or voice mailbox. In addition to higher client type prioritization for voice messaging (which can be considered much like application-level QoS), a voice-mail client has other needs that should be exploited through the TUI interface but supported through the voice-messaging

client interface in a given messaging system *if* they were to exist. These needs include fast retrieval of addresses when addressing messages and fast retrieval of spoken names for message addressing. Without these, subscribers experience slow response times in the TUI.

So, back to the present, it is necessary to provide a typical and (after a few years' maturation) simple definition of unified messaging: Using a single message store for voice, e-mail, and fax messages, a unified messaging solution can provide subscriber access to messages with the TUI or GUI in a consistent fashion.

This means moving voice messaging and all the functionality that subscribers are traditionally accustomed to essentially on top of a legacy e-mail system. Add fax on top of that, and you have a perfect solution. Achievable? Without a doubt. A realistic unified messaging solution certainly is viable and obtainable now.

So what does this mean? In essence, unified messaging is the convergence of voice, video, and data at the application layer, pure and simple. Saying this is probably too "worn" or repeated; nevertheless, the implications of unified messaging simply will not go away just because it sounds like an old idea. From this book's standpoint, this also means that Cisco Unity is a convergence product because it fully bridges the gap between legacy voice messaging and electronic messaging. To narrow or eliminate the gap between legacy voice technologies and data technologies is the whole idea behind Unity as a convergence product.

Unity as a Pure UM Product

In its original form, Unity was too pure of a UM product. Its original implementation was with Exchange 5.5. It used the Exchange 5.5 directory to store all its data and save a few pieces, and it used the Exchange 5.5 information store to store all messages. It did not store any data other than messages in the Exchange 5.5 store. The data that it stored on the local server consisted of the system prompts and default greetings, the call routing, rules, and the system schedule. Unity was installed with Exchange 5.5 on-box and was installed either as a self-contained system (which meant that it also had to be a domain controller) or as a member server in an existing Windows NT domain.

Some fundamental problems arose with this approach. The whole notion of using the Exchange 5.5 directory to store the UM-specific data required for the system to operate and for e-mail–enabled end users to act as Unity subscribers was not effective: The LDAP-enabled Exchange 5.5 directory was not fast enough to provide the real-time searching capabilities that Unity needed, and it often became unresponsive (the DAPI access was not any better). It was also very problematic for administrators, who had to worry about the impact of adding a large amount of data into the directory, challenging its directory size. Finally, the Exchange 5.5 directory forced replication for the entire object each time an attribute belonging to that object was modified. This meant that Unity caused directory replication to increase anywhere from marginal levels to very high levels, depending upon the actual subscriber traffic on the system.

Unity's use of the Exchange 5.5 information store represented a complete dependency on the capability of Exchange 5.5 to maintain a stable and operational store for e-mail clients and Unity alike. This dependency caused problems because it subjected Unity to frequent interruptions as the information store became impaired or unavailable.

To make things worse, Unity used the directory poorly, without regard for the response time from the directory and the way it depended upon message retrieval delays from the message store by playing back what it experienced into the subscriber's or outside caller's ear. As a result, the subscriber or caller heard all delays that Unity was experiencing—sometimes the delays were so bad that the system simply was not serviceable.

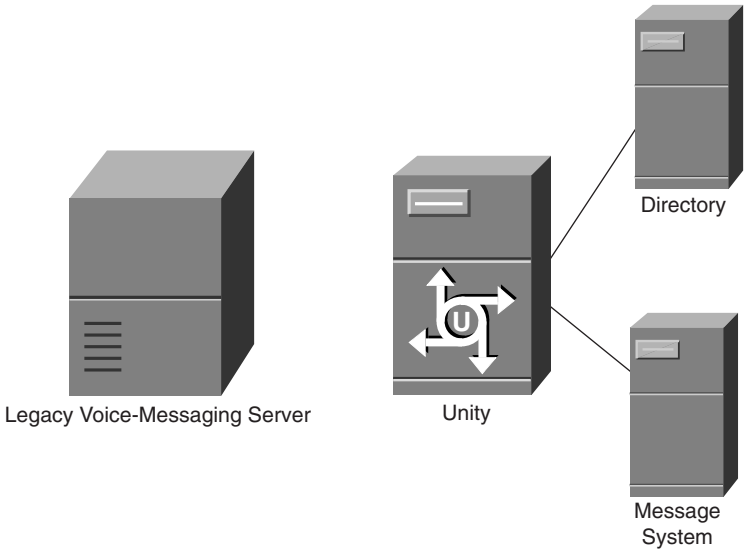
The challenge was to find ways to eliminate the response time issues. The directory was tackled first, so a proprietary cache was created in Unity that would cache the directory data. This made Unity capable of accessing the cache quicker than it could access the directory, making it faster and less susceptible to the inherent delays in accessing the directory directly. The information store came next: Continuous modifications and improvements were made in Unity's use of MAPI to compensate for the constant timeouts and loss of access resulting from the frequently unavailable or impaired information store.

As Unity matured as a product, and as Microsoft released Exchange 2000, more work was done to enable Unity to maintain sustainable operations. Thus, its pure UM implementation evolved to make it more reliable and to relieve its dependencies on the message store if it became unresponsive. The Unity cache, internally called the Dohcache (see Chapter 3, "Components and Subsystems: Object Model") was eliminated and replaced by MS SQL Server 2000. Using SQL was fully justified when it was determined that Exchange 2000's directory, Active Directory, was no faster than the Exchange 5.5 directory for the type of real-time directory access that Unity required. Thus, the SQL server implementation within Unity made the product more stable and more readily capable of providing a consistent end-user experience.

A part of this SQL server effort also included the notion of taking the information store off the Unity server, thus eliminating Unity's dependency on an on-box message store that could get bogged down by its connectivity to the other servers it communicated with. Thus, the standard implementation of Unity came with an off-box message store. This was perfect and desirable for unified messaging.

As a replacement product for a legacy voice-messaging system, Unity comes with a separation of account/directory functionality and message store functionality. Figure 1-1 shows the difference between a legacy voice-messaging system and Unity. In the figure, the main difference is that a legacy voice-messaging system contains its own mail store and some type of address book or directory. With Unity, those components are found in the customer's environment. Unity uses either MS Exchange or Lotus Domino. For specific versions, see Part II, "Deployment."

Figure 1-1 *A Legacy Voice-Messaging System Compared to a Unified Messaging Solution*

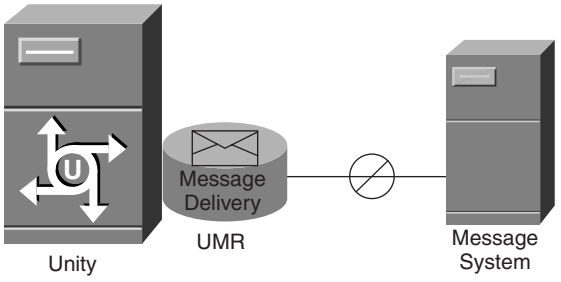


Unity Messaging Repository

The Unity Messaging Repository (UMR) was put in place because of the prevailing risk of losing access to the information store. The premise was that with the repository, voice messages would go into a holding place prior being delivered to the off-box Exchange server and stored there if the Exchange server or servers were offline. When a user's Exchange server is unavailable, the messages in the UMR are available to the user through the TUI, but not through the GUI interfaces. As soon as the messaging store comes back online, the messages are delivered and sent to the subscriber's mailbox. For more information, see Chapter 5, "Components and Subsystems: Messaging/Unity Message Repository."

Figure 1-2 shows Unity with the UMR in place. If it loses access to the messaging system that it services, it will continue to take messages from outside callers.

Figure 1-2 *The Unity Messaging Repository*



Comparison of Unified and Integrated Messaging

So, why stick with unified messaging if all these problems occur with the directory and the information store? Two very important reasons exist. First, you have just as many—although different—problems with integrated messaging.

With integrated messaging, you have the following:

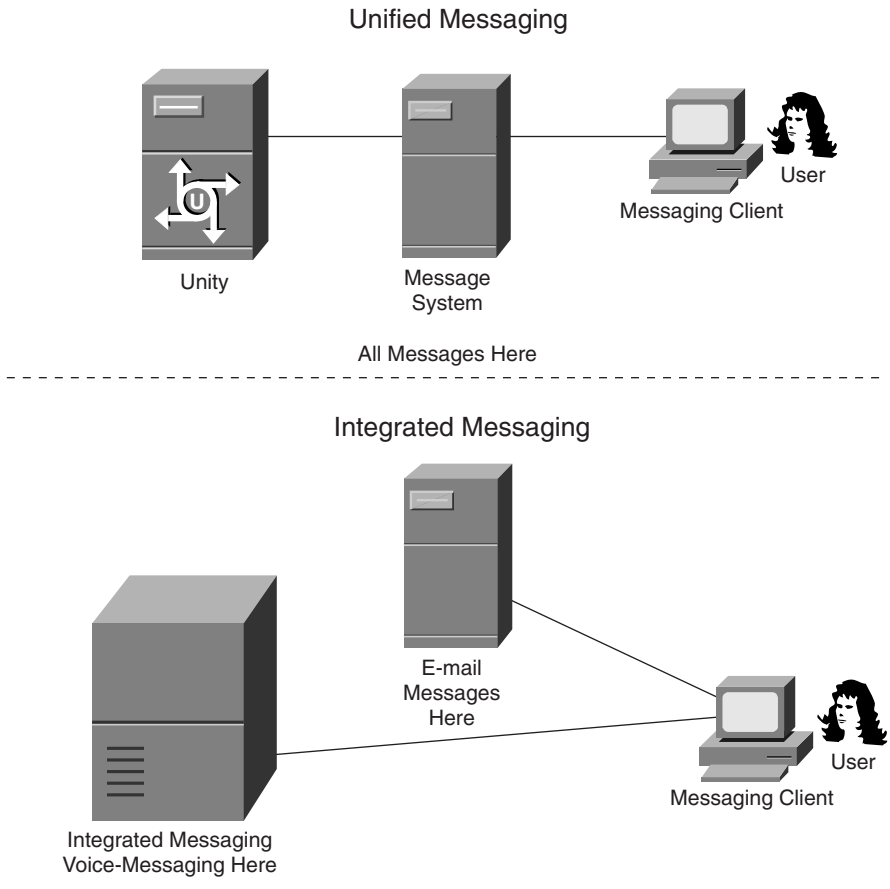
- Separate directories
- Separate message stores
- Client connection to both e-mail and voice mail (VM) systems
- Required VM address book when composing VM for an e-mail client
- No shared distribution lists

With unified messaging, you have the following:

- Same directory
- Same message store
- Client connected to the e-mail system only, but it can connect to Unity to use the Telephone Record and Playback (TRaP) feature for recording and playing back messages
- E-mail address book available when composing a voice messages for an e-mail client
- Shared distribution lists

Figure 1-3 shows the difference between unified messaging and integrated messaging. With unified messaging, the unified messaging server provides services to subscribers by becoming a part of the messaging environment where the subscribers reside. With integrated messaging, the integrated messaging servers become a second messaging infrastructure on top of the existing messaging infrastructure. The client then must connect to both messaging infrastructures to experience "unified" messaging.

Figure 1-3 *Unified Messaging Versus Integrated Messaging*



With integrated messaging, the focus moves off the messaging store onto the client, to unify the different types of messages. As a result, integrated messaging systems typically have a challenge in providing reliable notification services, such as lighting message waiting indicators, on a timely basis.

Challenges with Unified Messaging in an Organization

As with IP telephony, most businesses, especially Fortune 500 businesses, strongly desire unified messaging. For most organizations, moving toward unified messaging requires a lot of effort. This is of particular concern because of the legacy structure of most IS organizations and their prevalent separation of voice and data (for both technology and

the responsibility of managing it). For this reason, UM may not only be *challenging* to deploy, but it also could be entirely *impossible* to deploy. One interesting thing about this issue is that it has nothing to do with technical challenges or limitations of the product. Instead, it is mostly organizational. Many organizational issues arise in preparing a given company for unified messaging (although some technical challenges might exist for the organization's network and messaging infrastructure, such as readiness requirements and capacity planning needs). So, without understanding that a given organization's IS structure might prevent the deployment of unified messaging, little progress will be made in trying to deploy a unified messaging solution such as Unity within some companies.

NOTE Lack of organizational alignment toward UM is often a primary reason that larger companies choose to deploy Unity in a large-scale, voice mail-only configuration first and then deploy UM later: They want time to take advantage of Unity's technology and features, as well as the time to align organizationally to manage UM. This is fine and doable, but it also incurs a lot of extra work. A lot more work is involved because you must design a dedicated solution for Unity and then also take into consideration how you will migrate to UM. Occasionally, a migration to UM means moving data (subscriber information and possibly messages as well) off the voice messaging-only messaging systems to a newer version of the messaging system (such as migrating from Ex55 to E2K, or from Exchange to Domino). For more information, see the chapters in Part II.

After the organizational challenges are addressed, the technical challenges can be addressed. The organizational challenges actually might take considerably more effort than the technical challenges; it is very important to understand the issues surrounding organizational alignment—or the lack thereof. The following sections discuss the organizational issues that you should address before you deploy a unified messaging solution.

Who Manages the Messaging Topology?

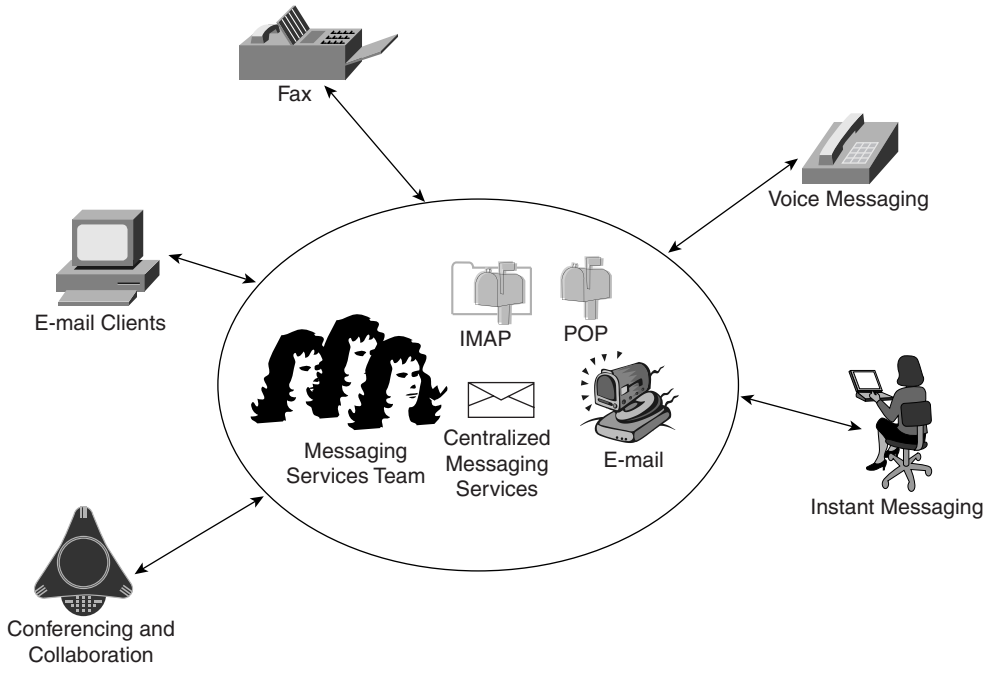
Organizations that want to deploy unified messaging must start with determining who internally manages the messaging topology. The messaging topology includes everything necessary for managing the messaging systems including dependencies. From the perspective of legacy voice messaging, the first response is that the voice team owns the voice-messaging solution. Is this still true with unified messaging? Absolutely. However, because UM focuses on centralizing different types of messages into the same messaging store, the voice team must actively be a part of the group managing the messaging topology.

Having the voice team join the e-mail team is a good start, although this is not perfect. An ideal scenario is one in which voice and electronic messaging skills, roles, and responsibilities are given to both teams as they join together. This might be considered just a certain

way to look at the situation, but the emphasis is important. Bringing both groups together or making both groups messaging-centric moves the emphasis from e-mail services and voice-mail services to messaging services.

Figure 1-4 shows a centralized messaging team with the responsibility for all aspects of messaging within a given organization. This includes voice messaging, fax, and e-mail—all parts of unified messaging.

Figure 1-4 *Organizing a Centralized Messaging Team*



So, why is merging the voice-messaging team and the e-mail team so important? It comes down to placing the emphasis on the core expertise required to maintain unified messaging, and it de-emphasizes the core competencies of separate e-mail and voice-mail teams. It is important to note that unified messaging puts new burdens on the e-mail infrastructure that require attention from the e-mail team. Unified messaging also builds a dependency between the voice-mail system and the e-mail system that the voice-messaging team needs to understand and manage. The benefits of merging both teams into one are numerous: Their joint expertise enables the effective support of unified messaging as a convergence technology. Many times, all team members must have IP telephony skills because when IP telephony is part of the deployment, the changes required can fuel changes to the voice-messaging legacy deployment and support team as well. Another benefit is that merging the

two teams' expertise prepares this same team for other types of messaging services in the near future, such as video messaging.

This is just one way that a cross-functional organization can be put together to properly support unified messaging. To maintain Unity as an application, the team must understand more than voice messaging: It also needs a core knowledge of the telephony integration methods that Unity uses. Within an organization that emphasizes messaging services—where voice messaging is just as crucial as electronic messaging—it is easier to manage the core competencies of the separate groups and allows them to share commonalities in their areas of responsibility. These common areas include second-tier end-user support, administration, and the common use of the messaging backbone and its directory service by all these different types of messaging.

Managing Perception Issues When Combining Voice Mail with E-mail

Another interesting paradigm to understand is how Unity and unified messaging affect organizations' typical perceptions of the differences between legacy voice mail and e-mail. It is often surprising to note that some companies expect and account for e-mail–based service interruptions on a regular basis. It might be scary to think that regular unscheduled e-mail outages are expected, but they are in some organizations.

In contrast, legacy voice messaging is perceived to always be available and isn't necessarily mission-critical. This seems to be a contradiction, but it might be the result of conditioning: A legacy e-mail system is typically liable to suffer service interruptions, and a legacy voice-messaging system never seems to have them.

One the one hand, for Unity to provide reliable voice messaging for subscribers and outside callers, it must always be available, even though it is not considered mission-critical. On the other hand, when Unity is installed in a messaging environment that is considered mission-critical yet unreliable, just knowing that Unity's service (normal operations) can be disturbed by an occasionally unavailable e-mail server can be enough to prompt the given company to uninstall the system before it even has a chance to provide unified messaging services to subscribers and before the company can realize its benefits.

If this is the case, what is the answer? How can Unity be deployed as a unified messaging solution if the messaging system that it connects to is occasionally unavailable? The answer is twofold. First, the perception must be changed. Reliability for any corporate messaging system (voice or e-mail) is essential to a successful and highly productive organization. So, if there is a concern about whether an electronic-messaging system can support unified messaging because the electronic-messaging system is unreliable, you must address those unreliability issues first. You must achieve the goal of reliability before you introduce Unity and unified messaging. Second, if achieving reliability in your messaging environment requires a redesign, include unified messaging requirements from the start. If it requires just

a correction of server hardware or OS, or the resolution of messaging application issues, make sure that Unity can operate reliably in this environment. Part II discusses in detail ways to assess and evaluate whether Unity can achieve normal operations with your messaging environment. Naturally, the emphasis on reliability with Unity includes two points:

- Ensuring adherence to deployment and design requirements found throughout this book—especially in the deployment section in Part II
- Focusing on a sustainable Unity solution, one where the loss of access to Unity's dependencies are minimized and Unity servers are placed in failover configurations

The messaging environment can and should be readied for Unity and should be maintained at that same high state of readiness, even after the product is deployed. In this case, a high state of readiness implies Unity's capability to maintain normal operations in an ongoing, persistent fashion. When the messaging infrastructure is readied, the next step is to understand Unity's relationship with and dependencies on this messaging infrastructure. In addition, it is important to understand how service interruptions, planned or unplanned, can affect Unity's capability to provide acceptable voice-messaging services in the same way as a legacy voice-messaging system that never goes down, as well as to voice-enable e-mail playback over the telephone. Ensuring this ongoing state of readiness is the primary way to prevent or minimize the chances of misperceptions of the product's reliability.

Back to organizational alignment to support unified messaging: Another very important argument for organizational alignment (aside from the need for centralized technical expertise) is that, without it, you are guaranteed to experience service interruptions with Unity. Of course, this creates undue perception problems about Unity's capability to provide reliable services to end users. It is easy enough to ignore Unity's messaging requirements for subscribers when it is trying to service their voice-messaging requests. Without properly aligning your messaging services, you are guaranteed to face serviceability issues between Unity and the messaging system that it is trying to service. This is another paradigm shift and is a prime justification for establishing a centralized messaging team that has both voice and e-mail expertise. One might say that this is the same paradigm shift, but so many technical and organizational issues are driven by these perceptions of unified messaging—and, in most cases, misperceptions—that they must be identified individually instead of being considered one in the same paradigm shift. So, because these paradigms are all interrelated, they must all be known, understood, and planned for if unified messaging is to be successful.

As an example, if your IS or IT organization is not aligned as previously discussed, it may not have all the information that it needs to work efficiently. If the e-mail teamtakes the messaging system out of service without telling the voice-messaging team to prepare for the same outage, you end up with a scheduled e-mail outage and an unscheduled voice-messaging outage. Both outages will affect the same end users, but in different ways. The users will know that the e-mail system will be unavailable during a certain period of time, but they will still expect to retrieve and leave voice messages and will be disappointed when

they cannot. So, without the organizational alignment, the end-user community suffers, propagating the perception that unified messaging is not suitable for its needs.

Another perception issue regarding unified messaging products in general is the expectation that features and functionality found in some legacy voice-messaging systems will transfer to unified messaging without any change in the behavior of those features or alteration of their functionality. This might or might not be the case. To address this concern and overcome the perception that this traditional functionality is lost, you absolutely must identify the functionality that is most critical for migrating to unified messaging. You do this by performing a usage analysis of your existing legacy voice-messaging systems and surveying your end users. In fact, end-user involvement is at the heart of a usage analysis: This analysis cannot be considered valid without end-user input and active participation in the migration to unified messaging. This usage analysis yields a considerable amount of data on everything you need to do to prepare for your migration. This includes what features of your legacy voice-messaging systems are most important to your end users, as well as a gap analysis between the legacy system and Unity. For more information on performing an end-user usage analysis, see Part II, especially Chapter 8, "Deployment Methodology;" Chapter 9, "Planning;" and Chapter 11, "Designing a Unity Solution."

An example of legacy voice-messaging functionality that is "lost" when moving to UM is deleting voice messages automatically that are reserved for a short period of time. Some of the most popular legacy voice-messaging systems offer a feature that enables users to store messages for a certain number of days before they are deleted. Subscribers depend upon this feature and use the system with an eye toward the eventual deletion of their stored messages. They know that the messages will be deleted and do not have to worry about maintaining their messages; thus, they do not have to worry about managing the size of the mailbox. This can be considered a nice convenience, but it is sorely lacking in unified messaging. For example, Unity uses a messaging system, such as Exchange or Domino, as its off-box store; it does not have the type of control necessary to delete old messages that have been left in a subscriber's e-mail box after a matter of days. Could it have that type of control programmatically through code? Absolutely.

Often an organization puts out an RFP stating that the UM system must delete voice messages after 12 or 14 or 16 days. At the same time, however, the organization does not want a product such as Unity to have that type of automated control over the messaging environment. So, it is possible to automate the management of voice messages in Exchange, for instance, but it is certainly considered too risky to enable Unity or any other unified messaging product to do this safely and without error. Most e-mail administrators are reluctant to enable any application to sit on top of their e-mail system and manage specific activities in this way, especially activities that include deleting messages periodically. So, even though it is desirable to have Unity delete messages after a certain number of days—and it would be useful to adhere to it—in the end, it is not practical for Unity have this type of capability within a messaging system, even if it is programmatically possible. Doing so gives Unity the liability for managing its subscribers' voice messages stored in the e-mail system.

Ease of installation and use are often differentiators between a legacy VM system and a UM system such as Unity. This major difference is the result of two primary reasons. First, most companies do not install their own legacy voice-mail systems. They typically have the maker of the product or one of their service's partners do the installation. With Unity, you must install it or have a Cisco partner install it. The second difference is that, during the installation operations, uncertainty arises concerning how the product actually should be installed. Unity has the same familiar, seemingly easy-to-use installation interfaces as other software-based products that use Microsoft technologies and APIs. However, you cannot just sit down and install it without having some knowledge of what the product is doing (or, in some cases, significant knowledge and access rights to the corporate directory and mail store). Thus, learning about what Unity does or is supposed to do is a key step in the installation process—and it is too often ignored. Do you want to stay out of trouble when installing Unity? Make sure that you know what it is supposed to do. Also do a few dry runs (in a lab) before you actually install it into your environment, regardless of the size of your environment.

Usage and New Security Issues

When voice messaging is introduced to the data environment, a whole set of new security issues arises. Understanding these issues and how to address them is crucial to a successful transition from legacy voice messaging to unified messaging. Ignoring these security issues and others like them will prevent you from realizing the finer benefits of unified messaging. It should be considered a best practice to address these issues during the planning and design process for any given Unity deployment:

- Privacy and confidentiality in voice messaging across an e-mail enterprise.
- Privacy and confidentiality in text to the speech of electronic mail through the telephone.
- Encrypted messages for the end user regardless if they're using their GUI e-mail client or the Unity TUI.
- Encrypted calls from Unity to CallManager and then from Unity to the messaging system it services.

These issues are presented in the following sections so that the awareness of them is raised from the start.

Privacy and Confidentiality in Voice Messaging Across an E-mail Enterprise

In a legacy voice-messaging system, messages do not have the freedom to travel in such an "out-of-control" way as what you might see with an e-mail message. Thus, a confidential voice message left for a voice-mail subscriber is heard only by that person—no one else. Not many options are available for forwarding confidential messages to just anyone, and

users do not have the freedom to edit the contents of the confidential voice message and resend it as if it came from the same original sender.

When voice messaging is introduced to a legacy e-mail environment, such as Exchange or Domino, confidentiality parameters must be addressed in the legacy e-mail or messaging environment. To prevent messages marked as confidential from being sent to just anyone, these confidentiality flags (marking a message confidential) must be used and maintained in your messaging environment. Without the support of and use of such confidentiality flags, voice messages can be sent to a wide number of people rapidly, without any capability to control who can receive the messages.

Privacy and Confidentiality in Text to Speech of E-mail Through the Telephone

With unified messaging, new functionality is present that does not exist in a legacy voice-messaging environment. Unified messaging has the capability to "voice-enable" a legacy e-mail environment, enabling the subscriber to play back voice messages *and* e-mail messages over the telephone. To play back e-mail messages over the telephone, text-to-speech (TTS) technology is used. This certainly sounds like a good idea, but what happens now that an outside caller can dial into a unified messaging system, log in as someone else—say, the CEO—and play back that person's confidential e-mail messages over the telephone? What happens is a very unhappy CEO.

Fortunately for Unity, it can support two-factor authentication that can then be tied to a class of service that supports TTS for subscribers. This means that subscribers who have the capability to play back their messages can do so only if they authenticate over the telephone using two-factor authentication. In Unity's case, this is the subscriber's extension and SecureID pass code entered from the subscriber's token or FOB. Without a pass code, Unity denies access into the system and prevents unwanted intrusion into mailboxes that have the capability to play back TTS. For more information about Unity's support for two-factor authentication, see the Unity Administration Guide on the Cisco website at www.cisco.com.

So, here is another paradigm shift. In a legacy voice-messaging environment, an intruder can call into the system and access the CEO's voice mail if that person can figure out his or her password. This has been an ongoing issue that seems to have been ignored or "played down" in its level of criticality to a business's daily operations. However, if you have a unified messaging solution and the same intruder accesses the CEO's voice mail, the issue is considered quite critical because the intruder also has a chance to listen to e-mail messages over the phone (if this feature is enabled). Both issues should be considered critical and they merit equal attention and care. In essence, by adding two-factor authentication capabilities to your unified messaging system, you alleviate both problems equally. As a subscriber, you must use your subscriber ID and SecureID pass code to access the system, whether you are checking voice mail or playing back e-mail. From an authentication standpoint, both are

now more secure. This means that, when it is applied, Unity's support of two-factor authentication for unified messaging is a far more suitable solution for playing back any type of message over the phone. If you will not use two-factor security, you can best keep your e-mail secure by not using TTS for subscribers to check their e-mail messages over the TUI.

Encrypted Messages, Encrypted Calls

If you resolve the issue of e-mail playback using TTS by implementing two-factor authentication, you take care of limiting security issues surrounding authentication. However, there are other forms of access, such as eavesdropping on the over-the-wire phone session. Thus, phones are not secure if the call between the phone and the switch that it is connected to is not encrypted. If all phone calls were encrypted, you would not have to worry about any eavesdropping: If eavesdroppers accessed the audio portion of the call, they would hear only the noise of encrypted audio.

Encrypted calls are out of the scope of Unity's capability. Nevertheless, they are always brought up as an issue when discussing message playback. Surprisingly, some organizations mention this issue during product evaluation more with message playback than with over-the-phone conversations.

Storing Voice Messages

The next question asked with any UM system such as Unity is, "Okay, so where are the voice messages stored, really?" The answer, of course, is that they are stored in the e-mail message store. This is actually a hard concept for people to accept. Are the voice messages stored somewhere on Unity? The answer is no. The voice messages are deposited in the Unity Messaging Repository (UMR); as long as the messaging system that Unity services is still online when the message is left there, it is delivered right away. A message stays on the Unity server only if the message system is unavailable. Should voice messages be stored on Unity or in both places, just in case? Possibly. They likely will be in the future—but not because Unity has to have any control over them. Unity eventually will need to consider its dependency on the messaging store to be essential but also behave in a benign way (that is, to minimize the impact the loss of this dependency on its normal operations) so that subscribers who simply want to dial in and check for new messages or manage old voice messages can do so, even if the message store is unavailable to them.

The important thing to note here is that Unity 3.1*x* and 4.0*x* can temporarily store messages (in the UMR) left by a subscriber or outside caller *if* the message cannot be delivered to an unavailable message store.

Another concern arises about storing voice messages with e-mail centers on the size of voice messages. Voice messages are created essentially as wave file attachments to an e-mail message. To play back the message, subscribers must use either the telephone or the

VMO snap-in for Exchange or DUC client for Domino in the e-mail client. These snap-ins for the e-mail client recognize the voice message and enable subscribers to interface with the voice-messaging system so that they can record voice messages from their e-mail client (GUI) and send to others.

If voice messages are just wave file attachments to e-mail messages, then how big are the messages? That depends on two things: the codec or codecs used to encode the voice messages, and the average size of the message. Does this mean that voice messages have an impact on an e-mail system? Absolutely! You cannot migrate your entire legacy voice-messaging system onto your e-mail system without affecting it in at least three ways:

- An increase in the number of messages
- An increase in the average size of a message
- An increase in the overall disk storage

Any unified messaging system also affects the load of your messaging systems—some worse than others. Unity's load on an existing messaging system varies, depending upon feature/function usage, the number of subscribers, and the number of messages submitted and retrieved. Be prepared to understand the effects that Unity has on your messaging systems. These effects are not a bad thing, but it is important to be realistic about the potential impact. They *can* become a bad thing if you are unprepared for them or don't manage them.

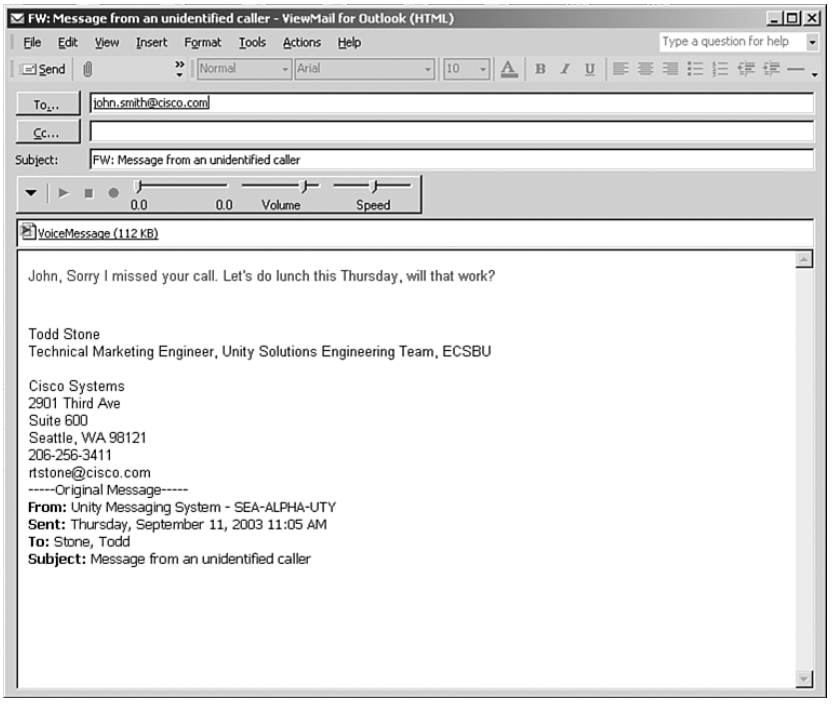
To learn more about Unity's use of codecs, see Chapter 7, "Components and Subsystems: Features;" Chapter 13, "Unity with Microsoft Exchange;" and Chapter 14, "Unity with Lotus Domino." To learn about capacity planning for your message store, see Chapter 9.

Remote Users, End Users, and Accessibility

Unified messaging represents a change for your end users. It represents a change in the way they communicate and interact with fellow coworkers or outside callers. With unified messaging, the end user now has the capability to review a voice message—say, from an outside caller—and then forward the message to that outside caller's e-mail address and respond to it using text (see Figure 1-5).

Having the capability to respond to voice messages using text or text messages using voice seem trivial and of no consequence. However, it is a means for end users to enhance or even improve upon their productivity, especially if it saves them time in communicating with one another.

Figure 1-5 *Responding to a Voice Message with Text*



With unified messaging, there is a contrast in conveniences for the different types of end users. End users who work in a corporate office every day will appreciate the convenience of unified messaging in their GUI interface. They can maintain a high level of productivity by staying at the computer and not being bothered with changing from computer to telephone to check for different types of messages. Telephone Record and Playback (TRaP) is the capability to use a media control in your messaging GUI client to record and play back your messages by having the control call your phone to use as a microphone or speaker. With this technology, a subscriber will still use a phone, but the phone is a part of the GUI experience. Of course, this means that users spend less time on the phone walking through conversations (listening to and responding to prompts played to them over the phone) and checking and responding to messages. In addition, end users can set their special phone settings, such as greetings and notification devices, through the Cisco Personal Communications Assistant (CPCA).

For mobile users, access to and use of the phone is essential, especially while they are in transit. With unified messaging, mobile users can receive and respond to voice messages, e-mail messages, and fax messages, and can maintain a higher level of productivity while they are out of the office. This benefit of unified messaging becomes important as usage increases over time.

Remote users benefit using either the GUI or the TUI, depending upon what is more convenient at the time. In some cases, especially with remote locations that have lower bandwidth, it might be more convenient for end users to use the TUI for accessing both e-mail and voice mail. When ample bandwidth is available, the GUI might be the only preferred method of access. In this case, both options are available to the end user.

All three examples show how different types of workers—office, mobile, and remote—can benefit from the use of unified messaging to improve their daily productivity.

Voice Mail and UM Coexistence

Another reasonable request is to integrate unified messaging with voice messaging. This just means that often it is desirable to have some subscribers use unified messaging and other subscribers use voice messaging only. So how is this performed? Because Unity uses an existing messaging system, it is necessary to configure the Unity unified messaging servers and the Unity voice-messaging servers to use the same e-mail organization instead of separate ones. For Exchange, this means the same organization name; for Domino, it means the same domain name. In this case, a messaging organization is the topmost messaging boundary established by the respective messaging systems when you first install them. The organizational boundary typically means the same address book and shared directory information.

With Cisco Unity, it is possible to have subscribers in one Unity server address messages to subscribers on other Unity servers. This is traditionally called digital networking; it is also called Unity networking.

This creates a challenge because now you have voice mail-only subscribers in your global address list. To keep unified messaging subscribers from sending e-mail to voice mail-only subscribers, it is necessary to hide their address in the global address book. Otherwise, they will receive e-mail messages that they cannot read.

NOTE Subscribers can play back e-mail over the phone by using text-to-speech. However, this is not considered a voice-mail feature and, therefore, is not available with a voice mail-only license. It is only available with a UM license.

So, although it is possible to mix voice mail-only subscribers and unified messaging subscribers within the same organization, the users must be separated on different servers because of licensing. To keep unified messaging subscribers from sending e-mail messages to voice mail-only subscribers, the voice mail-only subscribers must be hidden in the global address list.

Integrating Fax

Earlier versions of Unity contained a fax solution called Active Fax that was built into the Unity product offering as a separate server. Active Fax offered unified administration, and that was the biggest advantage of using it. Since Unity 3.0, Active Fax is no longer offered as a fax solution. However, Unity does have the capability to integrate with different fax vendors, depending upon the following:

- The offering is currently available on Exchange only.
- The fax vendor must use an Exchange message class to identify the message as a fax.

Solutions and Deployment

When you understand the paradigm shifts associated with unified messaging, you can begin to work on your solution and deployment. To develop your solution, you must understand how to define and execute the tasks necessary to transition or shift your organization to unified messaging. Then you must address the technical issues described throughout this book. When developing your solution, it is important to have an excellent understanding of Unity's capabilities. See the end of this chapter for a brief walkthrough of the different sections in the book, to make sure you understand the importance of each one. Play close attention to the Part II and Part III, "Solutions, Systems Management, and Administration," so that you can understand how to deploy Unity and also how to manage both it and your subscribers as you move to unified messaging.

Part II provides you with an in-depth look at how to plan for, design, and implement a Unity solution, regardless of the size of your deployment. Initially, it might seem complicated, but this will be easier when you understand how to identify the tasks associated with deploying Unity under various configurations.

Cross-Departmental Participation

One of the things you can do to address the seemingly large set of complicated issues is to give yourself an opportunity to openly identify the issues and features that are most important to your end users, regardless of where they are in the organizational structure. Some might frown upon this idea, but you simply cannot expect your end users to easily

accept something as different as unified messaging without involving them in the process. In a nutshell, it is easier for end users to accept the change that any new product brings with it, especially one as capable as Unity, if they can participate in the process.

It is possible to enable end users to participate and still allow them to be productive with their normal duties. Participation does not mean that they should be required to spend every working hour on the project. Aside from being directly involved, end users can participate by attending presentations and demonstrations of the product, by having access to the product in their area so they have a chance to understand what it does and how it does it, or by asking them to fill out surveys or questionnaires on the product's features and functionality. Other options include creating focus groups or working groups from different departments or divisions that can provide direct input into the product and how it might be used.

Adapting the User Community to Unified Messaging

So, how do you adapt the user community to unified messaging? when you involve users in the process, it is easier. You can do a few things to ensure that the end-user community has a chance to adapt to the technology:

- Offering training
- Easing in features over time
- Being aware of the changes in end-user behavior that might affect the system's performance (also known as the turnpike effect)

Training

Without a doubt, Unity's TUI conversation is different than others in the industry—no two conversations are exactly alike. Many of the key presses and conversation statements might be exactly the same or similar, but without including training in your deployment effort, you certainly will have significant support overhead. It might not be worth deploying a technology such as unified messaging if you cannot train your end users on how to be Unity subscribers. Make sure that you provide the necessary training for the end users.

Easing in Features Over Time

It is not necessary to turn on every feature or make every feature available to all end users at the time of deployment. As a matter of fact, you might find that doing so overwhelms the end users so much that they will not want to even use the product. Instead, limit the number of features and functionality. For instance, it is not necessary to grant every subscriber TTS

access. This can be managed by planning your class of service settings and making sure that some of your class of service groups do not have TTS access available. Another example is the web interface. Although it is functional, it might seem like too much to remember right from the start. Instead, give the subscribers some time to learn how to use the Unity TUI, and then phase in additional features over time. Doing so ensures a more pleasant end-user experience and makes your job easier in the long run. Your help desk will appreciate it as well.

Changes in End-User Behavior (the Turnpike Effect)

When trying to determine how to map features and functionality from your legacy voice-messaging system to unified messaging, consider a few important facts. First, to ease the transition, it is a good idea to attempt to closely match functionality between your legacy voice-messaging system and Unity. When you do, you can identify the gaps or differences in Unity's functionality with that of your legacy system. The gaps are areas where you will identify key training points for all end users.

In addition, end users might not be familiar with new features and functionality. These new features and functionality are the strict UM characteristics; they include technologies such as TRaP, TTS, and the capability to access personal settings (including settings such as notification devices and greetings) over the web. When subscribers begin using some or all of these features frequently, you will see a change in the performance of your solution. Thus, it is important to be aware of this up front so that you can plan for it. You can end up with a turnpike effect: End users might access the system more than they used to. So, at the beginning you might see the subscribers using the phone more often, or using the phone to record and play back messages using TRaP because controlling the phone from the computer is an exciting and new opportunity to use old technology. In this scenario, it is important to monitor port usage and call volume so that you can adjust accordingly. Again, as with all the critical issues listed in this chapter, do not ignore the possibility of the turnpike affect. Doing so can spell disaster for your new deployment.

Of course, you might find a complete change of habit over time. It is possible that phone usage will drop off drastically as subscribers become familiar with using the GUI interface to manage all their messages (see Figure 1-6).

Does this means that the subscribers will stop using the phone altogether? No, but it might mean that they will stop using the phone in the same way they used it previously, when the legacy voice-messaging system was in place. Figure 1-7 shows the difference between TRaP usage and legacy phone usage.

Figure 1-6 *Phone Usage Over Time*

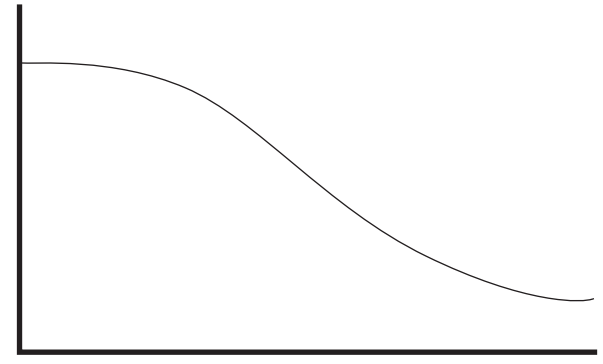
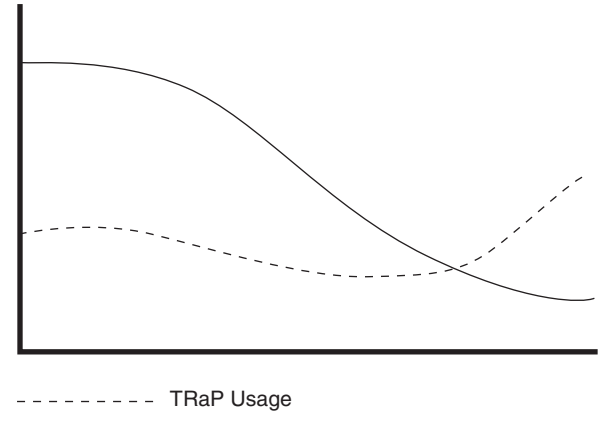


Figure 1-7 *Different Types of Phone Usage Over Time*



Administrative, Management, and Help Desk Considerations

Part II discusses how to plan for supporting your end users. The process that accompanies this planning is called an end-user feature/function analysis, and it includes identifying the feature differences between the legacy voice-messaging systems and Unity. This becomes the basis for support along with the new functionality that is provided to the subscribers over time.

On top of determining how to put a support structure in place, there are other considerations for administration and management of the unified messaging infrastructure. Most of these considerations are addressed in Part III, "Solutions, System Management and Administration," and include how to administrate multiple Unity servers and also how to do so programmatically. Take a look at the chapters in Part IIIto understand how you might build your support and administration structure around a new Unity deployment.

Messaging Technologies

Other chapters in this book discuss the messaging technologies that Unity uses; specifically, see Chapter 5. To give you a good idea of how Unity uses these different technologies, they are presented here briefly.

Unity's Messaging Technology

Unity's use of Exchange is based on the Messaging Application Programming Interface (MAPI). The MAPI implementation focuses on accessing subscriber mailboxes to retrieve messages on their behalf, sending messages on their behalf, and notifying subscribers when they receive messages.

The MAPI Profile and System Mailbox

With Exchange, Unity's key to servicing subscribers is through MAPI. That includes its use of a MAPI profile and system mailbox that is used to address messages from outside callers to subscribers of the system. The system mailbox also provides notification services for subscribers by monitoring their mailboxes for new messages and filtering on new voice messages. It basically monitors the state of all messages in a subscriber's mailbox so that it can act upon the state appropriately.

Unity's use of Domino is based on the Notes API; it uses the Lotus Notes client to access the Domino directory and messaging system. Through the Notes client, Unity provides unified messaging services to all subscribers. The implementation is a little different than in the Exchange version. This is largely the result of DUCS, which is in place to provide end-user proxy services for outside callers and to deliver messages to subscribers. The DUCS client provides a media master control for playing back and recording messages.

Integration Technologies

Unity was an early adapter of Telephony Application Programming Interface (TAPI) and made use of it for both its original voice board interface and its interface into CallManager. It uses TAPI through the telephony service provider (TSP), which is the interface connected

to the voice board or CallManager. The early voice board manufacturer used by Unity was Dialogic and is still in use today.

Unity's implementation into CallManager became more scalable and capable when it began using the Skinny Station Protocol in its TSP to connect to CallManager.

Along with Unity's support of multiple integration types, the Unity 4.0 offering includes a TAPI-independent implementation of the Session Initiation Protocol (SIP). In the future, the next-generation replacement of legacy voice boards used by Dialogic includes SIP as its primary interface. Unity will integrate with this interface for Dialogic. In addition, Unity can support third-party SIP proxy servers from different vendors.

A couple of very important chapters focus on these integration technologies: Chapter 6, "Components and Subsystems: Telephony Services," and "Chapter 17, "Unity Telephony Integration." In addition, you will find switch file settings in the appendix.

Summary

This chapter discusses the challenges associated with unified messaging as a part of the voice data convergence paradigm for the messaging application layer. It recommends that you pay equal attention to organizational alignment as to the technical aspects of migrating to a unified messaging system. It highlights several ways to address the different issues that the unified messaging paradigms present to organizations that are interested in adapting unified messaging as their strategic direction. It also discusses other important issues, such as how to prepare for unified messaging both technically and organizationally. Finally, it covers topics that end users might be concerned about as well, such as privacy and security.