Realtime publishers

The Shortcut Guide™ To

Secure, Managed File Transfer

sponsored by

IPSWITCH FILE TRANSFER

Don Jones

Introduction to Realtime Publishers

by Don Jones, Series Editor

For several years now, Realtime has produced dozens and dozens of high-quality books that just happen to be delivered in electronic format—at no cost to you, the reader. We've made this unique publishing model work through the generous support and cooperation of our sponsors, who agree to bear each book's production expenses for the benefit of our readers.

Although we've always offered our publications to you for free, don't think for a moment that quality is anything less than our top priority. My job is to make sure that our books are as good as—and in most cases better than—any printed book that would cost you $40 or more. Our electronic publishing model offers several advantages over printed books: You receive chapters literally as fast as our authors produce them (hence the "realtime" aspect of our model), and we can update chapters to reflect the latest changes in technology.

I want to point out that our books are by no means paid advertisements or white papers. We're an independent publishing company, and an important aspect of my job is to make sure that our authors are free to voice their expertise and opinions without reservation or restriction. We maintain complete editorial control of our publications, and I'm proud that we've produced so many quality books over the past years.

I want to extend an invitation to visit us at http://nexus.realtimepublishers.com, especially if you've received this publication from a friend or colleague. We have a wide variety of additional books on a range of topics, and you're sure to find something that's of interest to you—and it won't cost you a thing. We hope you'll continue to come to Realtime for your educational needs far into the future.

Until then, enjoy.

Don Jones

Copyright Statement

Chapter 1: How to Tell if You Need Secure, Managed File Transfer

If your organization is like most, you probably move more than a few files from place to place. There are probably several—if not hundreds of—vendors or business partners that you transfer information to, perhaps in an XML file or even a comma-separated values (CSV) file that was exported from a spreadsheet or database. Some of these file transfers may happen on a regular, scheduled basis; I'm betting that more than a few of your organization's file transfers happen on-demand, in a more ad-hoc fashion. You probably move files within your organization as well, either between departments or perhaps even between divisions.

One reason you may have started reading this book is the word "secure" in the title, although the word "managed" may have pulled you in, too. More and more organizations are grappling with that "security" word these days, either because they're simply tightening their own internal controls over their corporate information or because they're subject to industry requirements or legislative requirements that force them to secure and audit certain types of information. As more data flies back and forth across our private and public networks, we have good reason to become more concerned about who else might be reading that data—hence the focus on security.

If you've read the previous two paragraphs and thought, "Yes, this is my organization," then you've found the right book. I wrote this book specifically for organizations that need to move data from place to place, and need to do so in a secure, auditable, managed fashion, no matter what kind of data they're moving.

What Is "Secure, Managed" File Transfer?

This is a good time to get some terminology definitions out of the way. Many of the terms I'll be using in this and upcoming chapters are used in a variety of ways by a variety of different people, so I should spell out exactly what I mean by those terms so that you're not mislead.

File transfer is a generic term that applies to the act of transmitting data over a computer network—either a private network or a public one like the Internet. Notice that I wrote *transmitting data* rather than *transmitting files;* a file is really just a container for data, and it's not the container we're usually concerned with—it's the data within. In any event, any transmission of data more or less ignores the container—data is just a stream of bytes.

Managed is one of those vague terms that can mean a lot of different things. In the context of *managed file transfer,* it usually refers to software solutions that are designed to facilitate file transfer. There's even a commonly-used acronym: MFT. MFT solutions provide a "wrapper" around file transfer techniques and protocols, and that wrapper allows them to do things like schedule and automate file transfers, report on file transfer activity, measure file transfer performance and other metrics, and so on.

Secure file transfer is another overloaded term that people *think* they know the definition of. Secure file transfer is often a component of managed file transfer; the "secure" part usually refers to a bunch of specific capabilities, including:

- Encryption. This is what most people think of when they see the term "secure file transfer," and it refers to the ability to encode data in such a way that only the sending and receiving parties can view it.
- Auditing. This is an aspect of security that fewer people tend to think of right away, but it's an aspect that's becoming more and more important. It refers to the ability to track every activity associated with file transfer, such as who sent a file, when they sent it, who received it, when it was received, and so on.
- Non-repudiation. This refers to the ability of a file transfer system to ensure and prove that a file was received by the correct recipient.

There are other elements of security that we'll explore throughout this book, but this short list will do for now.

Obviously, you need software in order to achieve secure, managed file transfer; software has to provide the capabilities above and beyond those associated with simply streaming bytes across a network. In part, this book will be about the capabilities that you'll commonly find in MFT software solutions, so that you can do a better job of evaluating and selecting the right solution for your environment. If you don't really see yourself as a user of an MFT software solution, you may change your mind; Gartner has said that:

> Numerous factors cause companies to re-examine how they manage the movement of information from system to system, partner to partner, and person to person. FTP alone is not a viable option to give organizations the insight, security, performance, and, ultimately, the risk mitigation necessary to responsibly conduct business.

In other words, if your business is moving data, you probably *are* a potential user of an MFT software solution, whether you realize it or not. This chapter, in fact, will be a sort of test for you—if by the end, you're sure that none of these scenarios apply to you, then you're probably *not* going to be using an MFT solution, and you can stop reading this book (but wait until after Chapter 2, which is going to be really fun).

So now that we have a common vocabulary for secure, managed file transfer, where do you use it?

File Transfer Scenarios

Businesses typically have three scenarios in which they move data from place to place. It may seem a little redundant to talk about why you move data around, but this is actually an important place to start: *Why* you move data is where we'll find the specific business capabilities that you need. For example, if your business *never* transfers data on a recurring basis, then you may not need the same automation and scheduling capabilities that another business requires. So let's look at three basic scenarios as well as a couple of minor variations within each. Keep track of which scenarios apply to you.

Regularly Exchanging Files

The first scenario I always think of is moving data on a regular schedule between external business partners. The main reason my mind goes to this scenario first is that it has, quite honestly, been a major pain throughout much of my IT career. Years ago, I worked for a "dot com" that was a virtual retailer. In other words, we sold stuff that we didn't actually have. When we received customer orders, we transmitted those orders to the vendors that stocked those products, and the vendors *drop-shipped* the products directly to the customer. We had a very strong need, then, to regularly transmit order information to a huge variety of vendors, all of whom seemed to have different formats and protocols that I had to figure out.

Nowadays, my job would have been even more difficult. Because our customers invariably paid by credit card, we would have been subject to the Payment Card Industry Data Security Standard (PCI DSS), which outlines some pretty specific technical requirements for how we would have to handle customer data such as addresses, phone numbers, and so on. Our file transfers to vendors would have to be not only automated but also secured so that the customers' data wouldn't be revealed to anyone else.

I also had to worry about *receiving* files on a schedule, as our vendors transmitted invoicing information to us that way, although those didn't include any sensitive information. I not only had to have a place for them to send files—which would have been easy, because an FTP server would do the trick—but I had to watch for incoming files, grab them, and feed them off to a batch process that would import the invoice information into our accounting system. I needed more than just a simple FTP server, in other words: I needed a workflow-based automation system, or invoices wouldn't get paid and our vendors would soon stop dealing with us.

But external partners aren't the only instances when files are transmitted on a regular, recurring basis. In another job, I had to help coordinate the movement of data between an AS/400 and a Unix-based warehouse pick system. The AS/400 received sales information from hundreds of retail store locations and had to generate restocking orders for those stores. The restocking information had to be transmitted to the Unix warehouse system, which used a system of digital indicators to tell warehouse workers which products needed to go into which box for shipment to each store. Being a Unix system, it wanted mainly to deal with file transfer via common FTP, but we needed a very controlled, managed process to be sure the information got from one computer to the other. The Unix system would also transmit exceptions back to the AS/400—information on out-of-stock products, for example—so that the AS/400 would know that a particular store hadn't yet received a particular product and could be rescheduled for shipment when more product was received in the warehouse. The whole two-way transfer of data via a fairly simplistic protocol like FTP was a real nightmare in the beginning, and it's one of the things that first set me looking at MFT software solutions.

So this is the first of three scenarios: *Recurring, automated file transfer.* You'll also see this referred to as *system-to-system transfer* because data isn't being transmitted between individual people but is instead being transmitted directly from computer to computer. In addition, some automated processes are running on each computer to either produce the files being sent or to process the files being received.

Occasional or Ad-Hoc System-to-System Transfers

Another type of system-to-system transfer—again, not involving human beings—is the kind that doesn't occur on a regular basis but instead happens more ad-hoc, meaning that the parameters of the transfer are specified right when the transfer is made rather than in advance. I did this a lot, too, in former jobs. The dot com, for example, would sometimes need to add an extra set of orders for a specific vendor after our normal daily file transfer. That happened a lot during the holiday season when the order volumes were higher and the vendors had earlier cutoff times. The retailer I worked for would also need occasional ad-hoc transfers to the Unix system, as that's how we would download new warehouse maps and other data that didn't change very often.

Although these ad-hoc system-to-system transfers needed the same kind of security and management as the scheduled transfers, we found that we interacted with the file transfer system in an entirely different way. Rather than an administrator like myself setting up the transfer schedule in a back-end management console, we found we needed a user interface (UI) that a less technically-skilled user could operate. At the dot com, for example, ad-hoc late afternoon transfers were usually set up by someone in our order-management team because by that point in the day most of the IT staff was gone or were busy with other projects. At the retailer I worked for, new warehouse maps were created by the warehouse manager, and he didn't like having to wait on the IT staff to get to the file transfer—he wanted to be able to update that Unix system as soon as the new warehouse map was complete.

Another wrinkle was introduced when the warehouse manager started delegating the warehouse-mapping duty to one of his subordinates. He wanted *them* to set up the file transfer to the Unix system. However, before the transfer actually happened, *he* wanted to review and approve the new maps. So we had to somehow create a UI that would accommodate that business workflow: accepting a file transfer order but holding it until approval was received from a specific individual.

So the second of three scenarios, *ad-hoc system-to-system file transfer,* includes requirements for different kinds of UIs. In some cases, you may also have additional auditing or even workflow requirements.

Ad-Hoc, Person-to-Person Transfers

The last file transfer scenario involves people. Rather than transferring files from system to system, this scenario has people transferring files to each other. I did some consulting work for a hospital, and this was the most common type of file transfer there. Administrators would transmit patient records between departments within the same hospital and would transfer records between the hospital and external specialists like cardiologists, neurologists, and so on. Sometimes, they would do hospital-to-hospital transfers of records, when two doctors at different hospitals needed to consult on a particular patient.

You probably won't be surprised to learn that a lot of those transfers, at least when I started working with them, were done via email. In the next chapter, I'll spend some time explaining why email is a horrible idea for this kind of transfer. At the time, the hospital I worked with was just starting to implement their Health Insurance Portability and Accountability Act (HIPAA) requirements, and they had *just* figured out that email wasn't going to do the trick.

Ad-hoc, person-to-person transfers are tricky to deal with from a business perspective. You *have* to provide a way to accomplish them; otherwise, users will just use email attachments. If you restrict the size of attachments to make that option unworkable, they'll start using sites like http://drop.io, which is even worse.

What we eventually figured out is that the hospital needed a system that could essentially do "system-to-person" transfers of files, using full auditing, encryption, and the other fun stuff that HIPAA required. That system needed a simple UI so that administrators could easily initiate transfers from their desktops, feeding the required file or files into the system and letting it take over and actually send the file to the destination. In the end, it felt to the end users like a person-to-person transfer: They went to a Web page on their intranet, specified the files they wanted to send and the recipient, and the system took over from that point and made sure it all happened.

Ad-hoc, person-to-person transfers are the third major file transfer scenario that businesses see. Which of these three scenarios are occurring in your business?

Business Needs for File Transfer

People like us just need data moved from place to place; we often don't think much about the process beyond that simple requirement. But from a business perspective, there are a lot more things to worry about as data starts flying around on the private network or, even worse, flying around on the public Internet. In the next several sections, I'll discuss the details of the most common business needs that accompany file transfers. For each, I'll outline scenarios where I've run into that need in the real world and help highlight considerations that are often overlooked, even by large companies that have experience in this area.

Meeting Internal Requirements

These days, everybody likes to focus on "compliance" as the driver for all things security-related. Whether you're talking about PCI DSS, HIPAA, the Gramm-Leach-Bliley (GLB) Act, the Sarbanes-Oxley (SOX) Act, or federal government requirements, there seems to be no end of "compliance" requirements. And those examples are just in the US!

Companies have long had their own internal reasons for securing data. Simple internal compartmentalization is one reason: You might not, for example, want everyone in the company to have access to personnel records, salary charts, and so forth. In most companies, financial information and bookkeeping records are often considered confidential.

Concerns about corporate espionage also drive internal security concerns. The concept of *data leakage*—which is a really nice way to describe employees sharing data with those that they shouldn't—is always a concern, especially in companies whose business relies heavily on intellectual property that can't be adequately protected through mechanisms such as copyrights and patents.

Despite the growth in external requirements from legislative bodies and industry groups, internal requirements remain a strong concern for most corporate executives. File transfer is a key area in which data can be moved outside the organization, improperly disclosed, and used to potentially damage the business. Even accidental disclosure can be damaging, such as a case I had with one past employer. It actually wasn't the employer's fault; an employee improperly disclosed confidential personnel records for an employee that had recently been terminated. The terminated employee had applied for a new position with another company, and a person at that new company called a friend who still worked for ours, looking for information on their potential new hire. The friend dug around and provided that information—which resulted in the other company *not* hiring the person we'd fired. That person, of course, sued the heck out of us.

Our company could have defended itself against this by placing better controls on file transfer, but that's not really the first issue. After all, the information could have been divulged over the telephone or fax just as easily—except that both phone and fax *were* tightly managed, and using either would have created a trail of evidence leading back to the person who broke company policies about handling personnel records. If file transfers had been managed half as rigorously as phone and fax records, the offending employee could have been caught—and the company might have been able to deflect some of the legal damage onto the person who was actually responsible. As it was, we couldn't prove anything.

This is a big area where *managed* file transfer is intended to help: By not only placing some restrictions on who can send what, but most importantly by *auditing* what is sent, by whom, at what time, and to where. That audit trail can prove invaluable both for internal forensics as well as in legal defense, if it's ever needed.

Meeting External Requirements

I'm betting this is where you're expecting me to roll out the alphabet soup of industry and legislative requirements that we all refer to as "compliance," and I don't disappoint: HIPAA, SOX, GLB, FISMA, 21 CFR, PCI DSS, and more. The list is long and growing, but all of them have common general themes when it comes to securing and managing the transfer of files. They usually all require something like this:

- Data must be protected in-transit to prevent unauthorized disclosure, which usually means using encryption of some kind
- The transfer of data must be logged in a tamper-proof or tamper-evident log so that auditors can see who transferred what, when they did so, whom they sent it to, and so on
- Only authorized individuals should be able to access and transfer data
- In some cases, non-repudiation is required, meaning that there must be proof that the data was received by a particular system or individual so that the recipient cannot legitimately deny having received the data

But these are hardly the *only* external requirements that companies must deal with today. In many cases, external vendors or business partners may also set requirements for how *their* data must be handled. Go back to the dot com example I described earlier, and imagine that you work for one of the companies that we sent orders to. Those orders included customer information, and *we,* as *your* customer, had some requirements and expectations about how *our* customers' information would be handled: We didn't want that information transmitted to anyone else without our permission, and we wanted accountability for how that information was stored, accessed, used, and transmitted. Without assurances that our expectations would be met, we wouldn't do business with you.

In fact, customer expectations and assumptions are a major external requirement on data security. Let's say you stay in a hotel in a different country. In many cases, you'll be asked to show your passport when you check in, and the hotel may record your passport number, name, address, and other personal information. There's no worldwide rule on how that information must be protected, but you certainly *expect* that the hotel will keep your personal information under wraps, and you *assume* that they won't go sharing it with anyone inappropriately. You might use your credit card to pay for the in-room Internet access, and you would *expect* that information to remain private as well, even though that information might be shared by the hotel, the Internet service provider (ISP), a billing company, and possibly other parties.

I definitely had those expectations and assumptions when I checked into a hotel in Europe, and used my credit card to pay for the in-room Internet access. Several weeks later, when large, fraudulent charges started showing up on my account, I was a bit shocked. After some investigation, it turned out that the hotel collected my billing information for the Internet access and transmitted those files in batches *by unsecured, unmanaged FTP* to the Internet provider for archival purposes. Somewhere during that transfer process, the data was accessed and several credit card numbers "lifted" and used for fraudulent charges. Although neither the hotel nor the Internet provider broke any local laws, they certainly incurred the one penalty I could impose: I'll never do business with either of them again.

The fact is that many companies move all kinds of data from place to place, all the time. It's so commonplace that we barely even think of it; it's so easy in most situations that we definitely don't ever think twice. But simply moving data from place to place can be incredibly risky, and even if you're not violating internal company policies or legislative requirements, you may still leave yourself open to customers' wrath. That's one of the big reasons Gartner feels that MFT is such an important part of any business these days: We *need* to move files around, and we *must* do so in a secure, managed fashion.

High Availability

Let's take a break from security for a bit because it's hardly the *only* reason companies start looking at MFT solutions. High availability is another strong business driver for something better than simple FTP clients or email attachments; companies who rely on file transfers *need* their file transfer solution to be available all the time.

For example, let's go back to my dot com example. Originally, I had set up a bunch of FTP scripts on one of our servers to send drop-ship orders to our vendors. One day, that server stopped working. Nobody noticed because the server wasn't used for much else—it also had a bunch of archived product graphics and stuff, but nothing anyone had to get to continuously. In fact, it was a couple of days before we noticed it was down—and only because one of our drop-ship vendors called our sales manager to ask why we'd stopped sending orders every day. Oops. Obviously, we implemented monitoring solutions right away, but then I started thinking about it: Monitoring would tell me that there was a problem, but in our line of business, we couldn't afford for there to *be* a problem in the first place. What we needed was a set of *two* servers to handle file transfers so that if one broke, the other could take over. We eventually set up something like Figure 1.1.

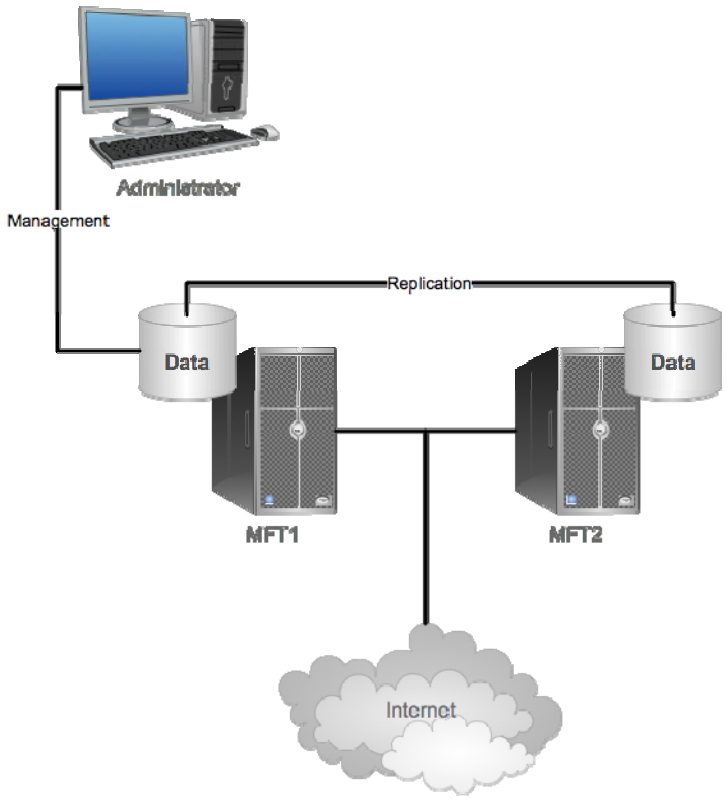

Figure 1.1: Highly-available file transfer.

Basically, we had two independent file transfer systems, each with a configuration database. The databases replicated with each other, so we only had to manage one of them and whatever we did would replicate to the other. The two coordinated, assigning jobs to each other so that they both had a roughly even workload. If one went down, the other would just pick up all the file transfer work. Incoming file transfer connections were balanced between them, so we'd have to lose *two* servers before we lost the ability to send and receive files. At first, we thought something like "load-balanced FTP clusters" were kind of ridiculous, but our CEO assured us it let him sleep better at night. Literally 100% of our business depended on incoming and outgoing file transfers; it was ridiculous, he said, that we had four load-balanced Web servers and only one file transfer server.

There are other reasons to create load-balanced, highly-available "file transfer farms." One might be to geographically distribute load. For example, if you have an office in the US and one in China, and frequently do file transfers within each continent, you might want to set up a server in the US and one in China to handle transfers within those continents. If the servers could be combined in some fashion, they could also offer failover for each other: File transfers from the US server to the Asian continent might not be as efficient, but it would be better than nothing.

Justification!

It's worth spending a little time thinking about what downtime in file transfer capabilities actually cost you. For my dot com company, it literally cost us tens of thousands of dollars in refunded orders to annoyed customers whose orders were delayed by 2 days. Knowing the *cost* of downtime will make it easier for you to balance the cost of adding high availability to your file transfer infrastructure; there are often many ways in which an MFT solution can be built for high availability, and knowing your cost threshold will help drive the necessary design decisions.

Communications Protocols

Businesses' file transfer needs are also strongly driven by the communication protocols they use to transfer files. In the ancient past of IT—say, 6 or 7 years ago—it was acceptable for companies to adopt proprietary protocols, forcing business partners to adapt to them. Today, with the wide availability of robust, open protocols, asking a business partner to switch to a different protocol is basically a slap in the face. The problem is that there are *so many* open, common protocols! Once a given business adopts one, they hate switching to something else, so in some cases, you have to be the flexible one, offering support for as many protocols as you practically can while meeting your other business requirements.

Today, the number of open, readily-available file transfer protocols is pretty large:

- AS1, AS2, and AS3. These *applicability statements* describe how to transport data securely and reliably. Security is usually based upon digital certificates and encryption. AS1 is based on the SMTP (mail transfer) and S/MIME (secure file encoding) protocols. AS2 is built around HTTP and S/MIME. AS3 utilizes FTP.
- Network file copy. This is simply an automated version of dragging files from a network drive to a local drive or another network drive, suitable for use within an intranet environment or over a Virtual Private Network (VPN). Common protocols for network file copy include Server Message Blocks (SMB, used by Windows) and Network File System (NFS, common on Unix-based systems).
- HTTP. The standard protocol for transferring Web pages between servers and browsers, HTTP is suitable for transmitting any kind of data. Web Services protocols (REST, SOAP, and so forth) utilize HTTP to transmit data, for example. HTTP is not intrinsically secured.
- HTTPS (HTTP over TLS). By adding Transport Layer Security (TLS) to a normal HTTP connection, you can add both encryption and authentication, helping to secure the entire connection and the data being transmitted.
- FTP. The granddaddy of Internet-based file transfer, FTP is generally quick and efficient, but it lacks any kind of intrinsic security, including encryption. It is extremely widely available, however, with FTP clients installed on virtually every kind of modern computer operating system (OS).

- FTPS (FTP over TLS*). This is an extension to FTP, adding TLS (often but incorrectly referred to as SSL). This is one of the most common forms of "secure FTP," a term that in practice can refer to several protocols.
- Secure File Transfer Protocol (SFTP). This is a part of the "Secure Shell" (SSH) protocol; it is also referred to as the SSH File Transfer Protocol. This isn't quite the same as running a normal FTP session within an SSH session (that's next), but is rather a completely unique protocol. SFTP is not to be confused with the Simple File Transfer Protocol, which is also sometimes referred to as SFTP.
- FTP over SSH. Yet another "secure FTP" variant, this protocol tunnels a normal FTP session through a Secure Shell (SSH) connection. This is also referred to as "Secure FTP." The actual FTP traffic is unsecured, but it runs through a secured, encrypted SSH session.
- Secure Copy Protocol over SSH (SCP over SSH). This works similarly to FTP over SSH, running a Secure Copy (SCP) session tunneled through an SSH connection. SCP normally encrypts transferred data, but the SSH tunnel also encrypts authentication and other traffic.
- SMTP/POP3. Email can be used to transmit data—after all, an email is really just data of some kind in a text format. Some organizations may send or receive data via email protocols, and SMTP and POP3 provide that capability.

These are the most popular and commonly-used open, Internet-based protocols in use today for file transfers. In fact, there are about a half-dozen other FTP variants (as if there weren't enough already), semi-proprietary file transfer protocols, and more. The list I've provided, however, contains the protocols that 99% of companies will be using for 99% of their file transfers.

To be frank, I consider almost all of these to be "must-have" protocols for a file transfer infrastructure. In any company I've ever worked for, either as an employee or a consultant, we've *eventually* needed to use almost all of these at one time or another. Even if we started out only needing, say, FTPS, we would run into other business partners who preferred SFTP, or a system that could only accept data via HTTPS, or a business unit that was receiving information via SMTP and POP3. It's become easier for me to simply specify *all* of these common protocols as base requirements, as that usually leads me to a solution that will last me longer, and serve in a larger variety of business situations.

* Secure Sockets Layer (SSL) is an older protocol that has been almost universally supplanted by the newer TLS. Both protocols work similarly at a high level, and referring to TLS as SSL is almost a habit with many technology professionals. In practice, the two terms are used interchangeably, although it is almost always TLS doing the work.

Programmability, Customization, and Workflow

I worked for a bookstore chain at one time. This was before Amazon.com really owned the book universe, and so we had plenty of brick-and-mortar competition. We didn't really compete on product; we all carried basically the same books, and could special order anything else a customer might want. We didn't even compete on price; publishers set the suggested prices for books and drive most of the promotions, so we all tended to have the same prices and discounts on the same books. What we competed on were our *business processes.* We worked hard to be better at stocking our stores with new titles, restocking old ones quickly, and so on. My point is that no two companies are identical, even if they're selling identical products at identical prices.

Because file transfer is so closely tied to business processes, you should therefore expect everyone's file transfer needs to be slightly different. That means a file transfer infrastructure has to work the way *your company* needs it to—not in some generic fashion that your company has to adapt to.

There are many ways in which MFT solutions accommodate different business processes. Some solutions offer an Application Programming Interface (API) that allows your own software developers to create custom file transfer consoles and clients or to incorporate file transfers into line-of-business applications and other custom business processes. These APIs may work for Microsoft's .NET Framework, Microsoft's Component Object Model, Sun's Java, or some other development platform. Some vendors may offer a variety of APIs to accommodate a variety of development languages.

Vendors might provide a custom scripting language, or support existing scripting languages, so that you can "program" your MFT solution by writing simpler scripts and batch files. Others might provide command-line utilities that can be scripted by an experienced administrator or programmer to customize specific operations to meet the company's business processes.

Another way in which MFT solutions can be customized is through *workflow.* Typically, this involves much less expertise and overhead than programming. In addition, this option offers a solution to situations in which one person may queue up a file for transfer but another has to review and approve it, and you want to track all that review/approval activity in a log of some kind. Workflows within your business may be simple "review/approve" workflows or they may be complex processes that mirror specific business processes that have been defined within your organization. Solutions do vary considerably in how they allow you to define these workflows: Some may actually require some level of scripting or programming, while others may use graphical UIs (GUIs) to let you visually connect workflow components into a complete process. I'm definitely a bigger fan of the "graphical" style of workflow construction, as it allows less technically-skilled users—such as business process owners, rather than programmers—to construct workflows and even maintain and modify them on an ongoing basis.

Integration with Existing Technology Assets

Your business likely already has a significant technology investment, and adding a formal file transfer infrastructure shouldn't require you to re-build much of that infrastructure. Ideally, an MFT solution should integrate well with other technology assets, such as existing Web sites for data transfer or directory services for user authentication.

There are a number of potential integration points that you can consider for your file transfer infrastructure:

- Directory services. Whether it's Active Directory (AD) or some other directory, you might want to have your file transfer infrastructure authenticate users from an existing directory services. More and more organizations are seeking to reduce their total cost of identity and access management (IAM), and solutions that utilize an existing directory—rather than adding another user database to the environment—help support that cost reduction.
- Databases. A file transfer infrastructure requires a database to store configuration, job, schedule, and logging information; the ability to use *existing* database resources—such as an existing Microsoft, Oracle, or IBM database server—can be a benefit to some companies. Using an existing database means the file transfer solution will add less administrative overhead to the environment. However, some MFT solutions are entirely self-contained, using their own internal database. Provided that internal database doesn't require excessive maintenance and administration, it may not add enough overhead to worry about.
- Web servers. Some MFT solutions offer Web-based UIs, especially for users authorized to create ad-hoc transfers. Figure 1.2 shows an example of a Web-based interface. Some solutions may be able to expose this interface through existing Web servers; others may have an embedded Web server that requires no additional maintenance. Less desirable are solutions that require you to add a specific new Web server that you otherwise wouldn't need, such as adding an Apache server to an all-IIS shop or vice-versa.

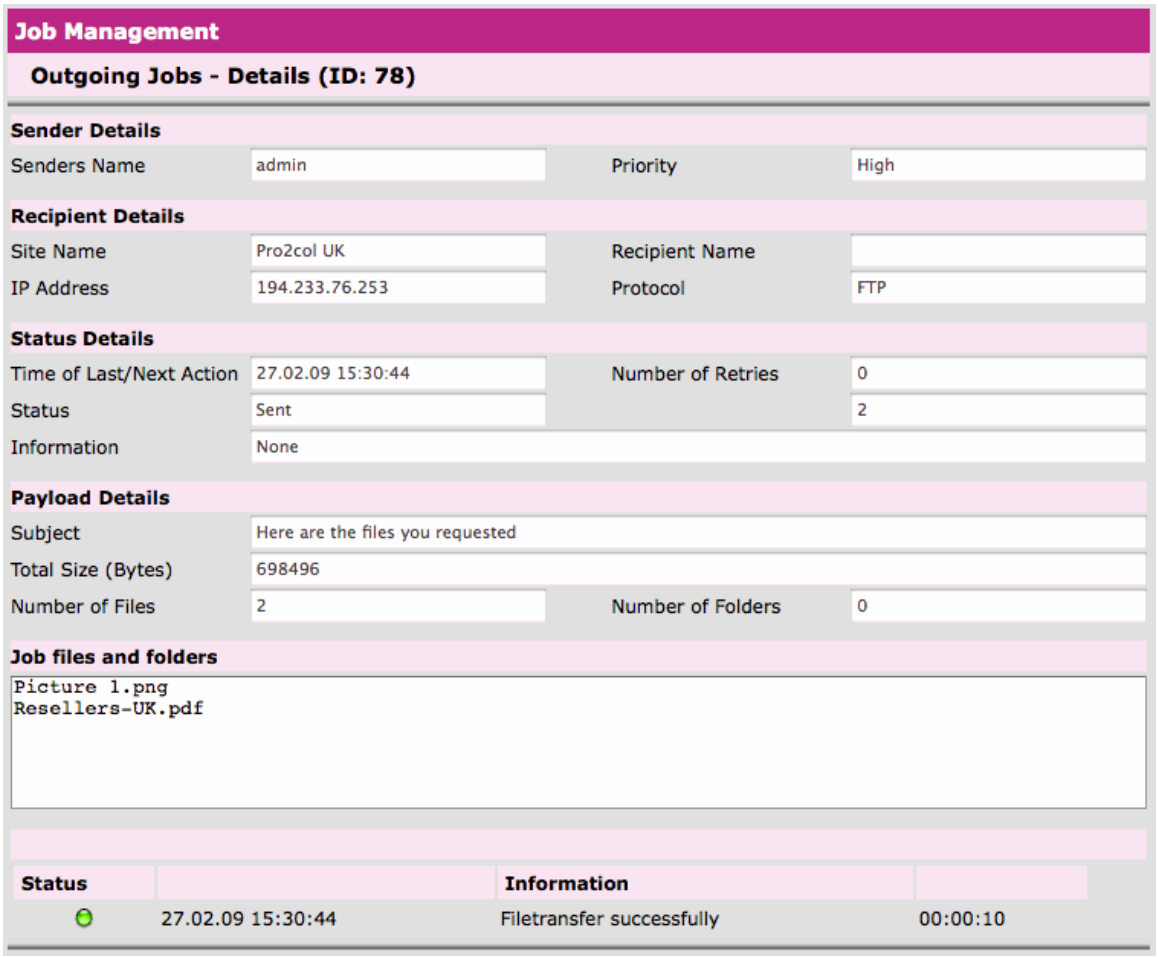

Figure 1.2: Web-based file transfer UI.

Other integration points might relate to specific line-of-business applications, such as predefined workflows that integrate with inventory systems, customer management systems, and so on.

Scheduling and Monitoring

Scheduling is important for recurring file transfers, of course, but it can also be important for ad-hoc transfers, either from system to system or person to person. In other words, just because I want to set up a one-time file transfer doesn't mean I want it to happen *right now;* I may need it to happen later in the day or even on a specific day in the future. Your scheduling needs may be fairly simplistic or quite complex. Figure 1.3 shows what a file transfer infrastructure might offer in terms of a fairly simple, straightforward UI for scheduling one-time or recurring transfers.

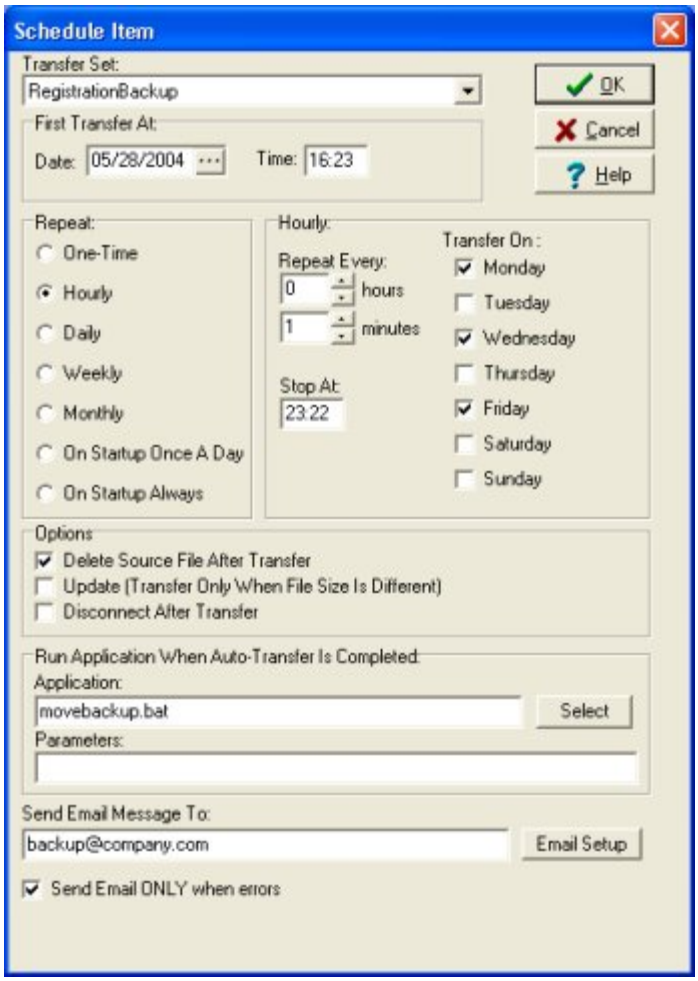

Figure 1.3: UI for scheduling transfers.

Monitoring really falls into two broad categories. The first is the ability of individual users to monitor their file transfer jobs; Figure 1.2 showed how a simple user-oriented interface might do that. Other systems might offer users the option of receiving status emails when their jobs complete, run into a problem, and so on.

The other category of monitoring is the broader, IT operations-level monitoring. IT needs to be able to monitor the health and performance of the file transfer infrastructure, receive alerts when something goes wrong, and so on. This monitoring might be accomplished by a console or utility specific to the MFT software that you implement. In other cases, a solution might provide monitoring that integrates with other monitoring consoles, such as HP OpenView, IBM Tivoli, or even Microsoft's System Center Operations Manager. In still other cases, the solution might simply expose monitoring *instrumentation,* such as Windows performance counters, which can in turn be accessed by a wide variety of monitoring tools.

Both types of monitoring are useful and desirable. Users will be more likely to use a file transfer solution if they don't feel that they're dumping their transfer requests into a big, black box; being able to check the status of their jobs and receive notifications adds a level of confidence, and confidence leads to usage. Operations-level monitoring is obviously crucial for any mission-critical service; IT needs to be able to spot upcoming problems based on patterns and trends, and needs to be alerted quickly if something fails or goes wrong.

File Transfer Frequency and Volume

This is a tricky business requirement that I think a lot of people overlook—I certainly did when I started dealing with automated file transfers back in the day. In fact, there's a good story here. I used to work for a network engineering and management company, which was a sub-division of a regional utility company. To make a long story short, we offered our customers the ability to have our services appear on their utility bill—much as you might pay for satellite television on your telephone bill today. Actually making that happen was a *lot* harder than you might think: We not only had to get our data into the right form but we had to transmit it in a *very* specific fashion, at *very* specific times. We didn't realize how specific those times were until, one month, we transmitted everything a couple of days early to work around a holiday.

It turns out that the reason our delivery schedule was *so* specific was that the utility's file transfer server was single-threaded—it could only handle one file transfer job at a time. Crazy, right? And the day we decided to send our billing information happened to be the day that another subdivision was assigned—and so we basically crashed the whole system. Oops.

The moral of the story is this: Think about the frequency and volume of your file transfers, and build your file transfer infrastructure appropriately. Find out if there's a limit on the number of simultaneous tasks, for example, especially if you'll be implementing complex workflows for file transfer. Get a feel for the maximum sustained data throughput your proposed infrastructure can support, and decide whether it'll be sufficient for your purposes. If you'll be handling a truly enormous amount of file data, you may even need to consider load balancing within the file transfer infrastructure. That load balancing can also provide a degree of high availability, helping meet those two business needs.

Content Security—Preventing Malware

There is a ridiculous amount of malware out there today, and a file transfer infrastructure offers a huge opportunity for more of it to enter your environment—and an opportunity for malware *in* your environment to spread to *other* environments. Fortunately, *most* of what a file transfer infrastructure handles are simple data files—CSV files, XML files, and so on—that don't contain any executable code. But *some* files that move through your infrastructure *will* contain executable code, and you need to deal with it.

My preference is to *not* have the file transfer infrastructure implement its own anti-malware measures. I have more than enough anti-malware software in my environment that needs to be kept updated; I hardly need another. What I prefer to see is a file transfer infrastructure that can *use the anti-malware stuff I already have.* In some cases, that may mean simply dumping files to disk where my anti-malware software can scan it like it does every other new file. Higher-levels of integration may allow file transfer infrastructure components to actually submit incoming files to the anti-malware engine before writing it to disk or doing anything else with it.

Another scenario is a file transfer infrastructure that doesn't integrate with the specific anti-malware solution you have but *does* integrate with an existing, third-party, well-known anti-malware solution. I'm okay with that, too, in large part because it brings another malware-scanning engine and technique into the environment, meaning my total anti-malware effort is more likely to catch *everything.*

Cost

This last business driver is hardly ever the least important: How much will it cost? All of the other business concerns are—and should be—weighed against the cost. If high availability costs significantly more than what would be at-risk for *not* having high availability, then you don't get high availability.

Ideally, a file transfer infrastructure should be somewhat modular. Businesses shouldn't be forced to buy a "one size fits all" solution, because no business is exactly like another. Modular components—making high availability an option, for example—help business customize a solution that fits their needs and risk mitigation requirements in a cost-effective fashion. When I'm building any kind of infrastructure service, including file transfer, I like to look for solutions that offer just one or two feature-related "editions," and then lots of "options." That way, I can build what I need now, and add options later as I need them and can afford them. It's a bit like buying a full-sized computer over a laptop: With the laptop, you get everything in one package, but you have to be careful to buy the best one you'll ever need because they're relatively difficult to upgrade. A full-sized computer, however, can usually be opened up, and its components can be upgraded, swapped out, and so forth, all with relative ease. That means you can buy a relatively low-powered computer to begin with, then upgrade specific options as needed in the future.

Perhaps most important, though, is something I'll address in more detail in the next chapter: The cost of *doing nothing.* In other words, if you have file transfer needs, you can't just consider the cost of a file transfer infrastructure as overhead. Why?

If the business truly needs something, that something will get accomplished somehow. If it isn't through a formal file transfer infrastructure, then it will be through an *informal* infrastructure, often composed of cobbled-together components, do-it-yourself scripts, and so on. Those *are not free.* It can be very difficult to discern their true cost, but there is—as we'll see in the next chapter—a cost in maintenance, risk, and so forth. Compare *that* with the cost of a more formal, integrated, supported file transfer infrastructure that meets *all* your business needs.

Coming Up…

This chapter has been all about the reasons you might need secure, managed file transfer. Hopefully, you've seen some of your own business needs in the ones I've outlined here—and you're at least starting to see yourself as a potential user of secure, managed file transfer. In Chapter 3, I'll help you start mapping these business needs to the technological capabilities that you'll find in various file transfer solutions in the marketplace so that you can start building an evaluation checklist. However, before I do that, I want to debunk common myths about file transfer—and that's what's coming up in Chapter 2. There are definitely plenty of myths to look at: that security isn't important, that you can build a homegrown solution more cheaply and easily, that all kinds of security—like encryption—are basically the same, and so forth. We'll put these to the test, and figure out which common file transfer myths actually stand up to rigorous, logical scrutiny.

Download Additional eBooks from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this eBook to be informative, we encourage you to download more of our industry-leading technology eBooks and video guides at Realtime Nexus. Please visit http://nexus.realtimepublishers.com.