Realtime publishers

The Shortcut Guide™ To

Eliminating Insecure and Unreliable File Transfer Methods

2012 Edition

sponsored by

Attachmate

Dan Sullivan

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: Understanding the Limits of Traditional File Transfer

If you had to identify the most salient characteristics of enterprise IT operations, you would certainly include the enormous volumes of data in motion. Data moves from point of sales systems to back-office systems, from financial systems into data warehouses, and from data centers to failover and disaster recovery sites. This data moves through IT infrastructure in a number of ways. Tightly coupled systems may use application programming interfaces (APIs) to pass individual, transaction-level data. This is a sound solution when requirements demand rapid movement of data as soon as it is available; for example, when passing credit card transaction data to a risk analysis system that predicts fraudulent activity. In many cases, however, data is more efficiently moved in batches from one application to another. In these cases, file transfer is the tried-and-true method that is widely used.

The Shortcut Guide to Eliminating Insecure and Unreliable File Transfer Methods examines how common practices in file transfers undermine the efficiency and security of business operations. This guide consists of four chapters, each of which addresses a particular aspect of the file transfer problem and offers a way of mitigating those problems.

This chapter starts with an examination of the question: What is it about common file transfer methods that are so problematic? After all, if so many of us have been using these techniques for so long, how bad can they be? The unpleasant answer becomes clear pretty quickly once you start delving into the details. In brief, the shortcomings you can tolerate in one quick-and-dirty file transfer solution can easily find their way into multiple mission-critical workflows undermining the integrity of these broader systems.

Chapter 2 considers business objectives that are jeopardized by the problems highlighted in Chapter 1. These include compliance, flexibility, and scalability of IT infrastructure, management issues, cost control, and workflow efficiency.

By the end of Chapter 2, you should have a clear understanding of the problems with insecure and unreliable file transfer methods as well as the business issues that are driving the need for a better solution. Chapter 3 defines functional requirements that can help guide the selection of a secure, reliable, and efficient file transfer system. This chapter considers the need for supporting information flows, security, file transfer optimization, transaction controls, process reporting, and event processing.

Chapter 4 moves on to issues related to deployment and management. Once a secure and reliable file transfer system is in place, there will be questions about policies, procedures, and service level agreements (SLAs). Chapter 4 walks through a deployment methodology that begins with assessing current file transfer methods and identifying critical business requirements to establishing policies and procedures and rolling out a file transfer solution.

Together, these four chapters will help you understand the limits of traditional file transfer solutions and the risks they pose to businesses. This guide will also provide guidance on selecting, deploying, and maintaining a secure and reliable alternative file transfer solution.

Characteristics of Typical Homegrown File Transfer Solutions

Many IT professionals have had to deal with the problem of transferring files. Software developers routinely work with files, generating output from their applications that will be needed as input to other applications. Systems administrators are constantly working with operating system (OS) and network log files, application output files, backup files, and a seemingly endless list of other types of data files that are needed in today's enterprises. As these examples may indicate, file management is often part of a larger workflow and that, itself, can be a problem.

Perils of Quick-and-Dirty File Transfer Solutions

Let's consider a hypothetical scenario involving file transfer tasks. A sales manager in a company has determined that online sales are underperforming expectations. She thinks the problem may be related to usability issues in the Web site design, so she has asked the business intelligence (BI) analyst in the group to analyze click stream data to better understand the paths customers take through the Web site.

The BI analyst decides to pull log data from the Web server and run it in his favorite statistical analysis package. He'll need to keep his copies of click stream data up to date, so he decides to write a Perl script to copy the file from the server to his desktop. After a few days analyzing the data, the BI analyst develops a program to identify patterns in the click stream data and use them to classify several types of customer interactions or sessions with the Web site. The sales manager finds the analysis is just what she needed and decides to incorporate the results into the sales data mart.

The data mart has evolved over the past few years in the sales department, usually by incremental additions of data. Some source data comes from a file system, some is extracted from other databases, and a couple of other sources are on an application server and another Web server. Adding the click stream data is just another data source, so past patterns are followed. An extraction, transformation, and load (ETL) workflow is created to copy the statistical analysis results from the analyst's workstation to the data mart. (The sales manager knows that process should really run on a production server, but there is no time to do the migration now; all involved agree to get to that as soon as possible.) The result is a workflow depicted in Figure 1.1.

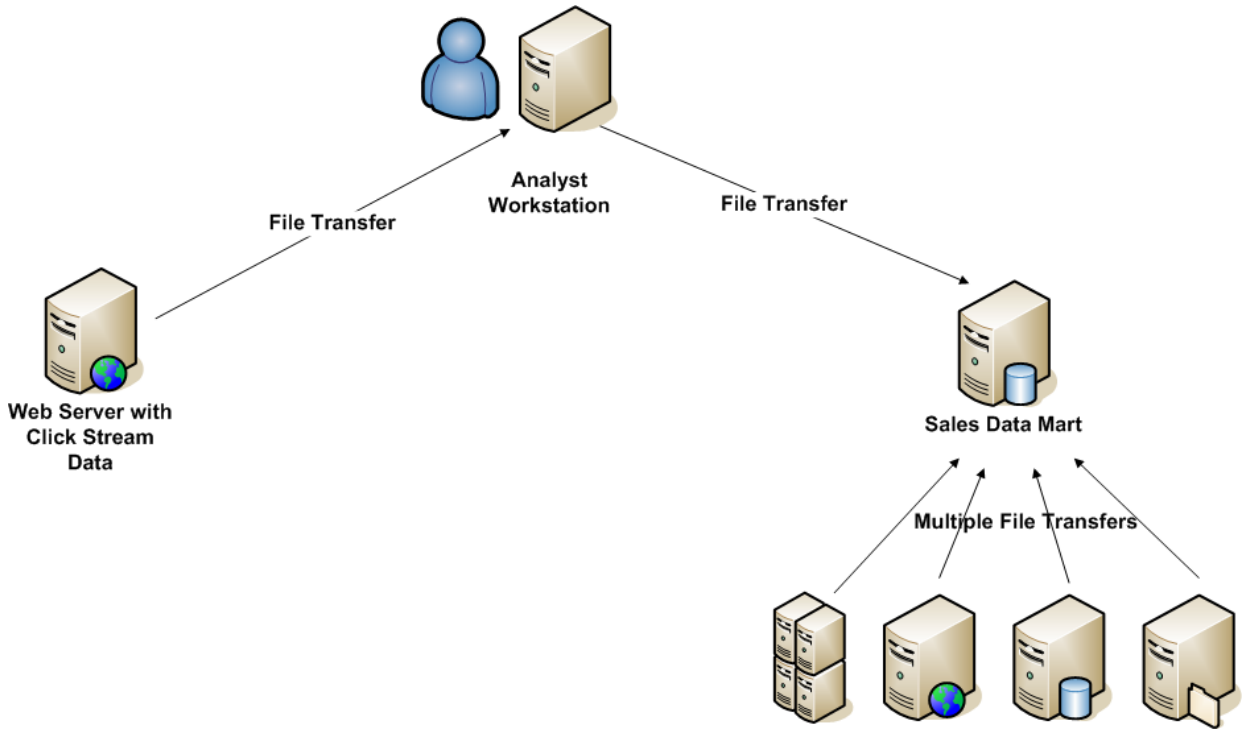

Figure 1.1: Simple, isolated file transfers can quickly become critical parts of larger workflows.

What Figure 1.1 does not show is how the different file transfer processes are implemented. In this scenario, there are several methods:

- The BI analyst is an adept Perl programmer, so the file transfers from the Web server to his workstation and from his workstation to the data mart are performed with Perl scripts he wrote specifically for those operations.
- The developer responsible for transferring files from the file server, which is running a Windows OS, used some Visual Basic code from another application and adapted it to this file transfer problem.
- The files from the database server, which runs Unix, are copied with Unix commands in a Korn-shell script.
- The files from the Web server and application server, which both run Linux, are preprocessed with text processing utilities, awk and sed, before they are copied by Perl scripts to the data mart.

Each time a need arose to transfer files to the data mart, the developer responsible for the task took the most efficient path: writing a custom script to do exactly what was needed using a programming tool that best fit the source platform of the files. As a result, a series of local, optimal decisions leads to a substantially sub-optimal global solution.

File transfers are such a common and fundamental task in IT operations that the task warrants a global solution. To understand what that global solution should include, let's consider in more detail some of the characteristics of homegrown solutions.

5 Detrimental Characteristics of Homegrown File Transfer Solutions

Five characteristics stand out when considering homegrown file transfer solutions; each of these contributes to undermining their reuse and maintenance. These characteristics are:

- Custom scripts
- Focus on a single problem
- Duplicate functionality
- Limited error handling
- Difficult to maintain

These characteristics are closely related and collectively undermine the long-term utility and generalized use of these programs.

Custom Scripts

File transfer is one area where a common need frequently leads to individual solutions. Many of us in development and systems administration roles have written custom scripts for managing file transfers. It is so commonplace, many of us would not think twice about starting up our favorite editor and cranking out the Perl code to get the job done. In the best of cases, we use design patterns and common programming constructs along with copious documentation so that the next person who comes along to maintain the script will not feel like she is deciphering Mayan hieroglyphics.

The biggest problem with custom scripts is the need to keep recreating a solution to a common problem. This is unfortunate. Other areas of software engineering and systems management use common solutions, including OSs, databases, application servers, report generators, and other problem-specific applications. Custom scripts are by definition, not standardized; they are:

- Written in different languages. Perl is probably the most popular programming language used but Python, Ruby, and Unix shells are useful for file transfer operations.
- Written by different developers. Perl's motto is "There is more than one way to do it." Perl is a versatile language and that leads to multiple solutions to common problems, such as file transfer. Design patterns are available for data manipulation in Perl (see, for example, David Cross' *Data Munging with Perl*, Manning Publications, 2001) but when under the gun to deliver a solution, developers understandably turn to methods and techniques they have used and know work.
- Written to solve a single problem rather than abstract the problem and solve the more general problem

The reason for such customization is that these scripts tend to focus on a single problem.

Focus on a Single Problem

A second common characteristic of homegrown solutions is that they tend to focus on a single problem. This focus stems from the localized view of the problem at hand. Developers on a project are charged with completing their development on time and in such a way that reliably meets the project requirements. Similarly, systems administrators are responsible for keeping applications and systems up and running as efficiently as possible. These perspectives do not lend themselves to solving the more global problem of creating a robust, reliable solution for file transfer tasks across the enterprise.

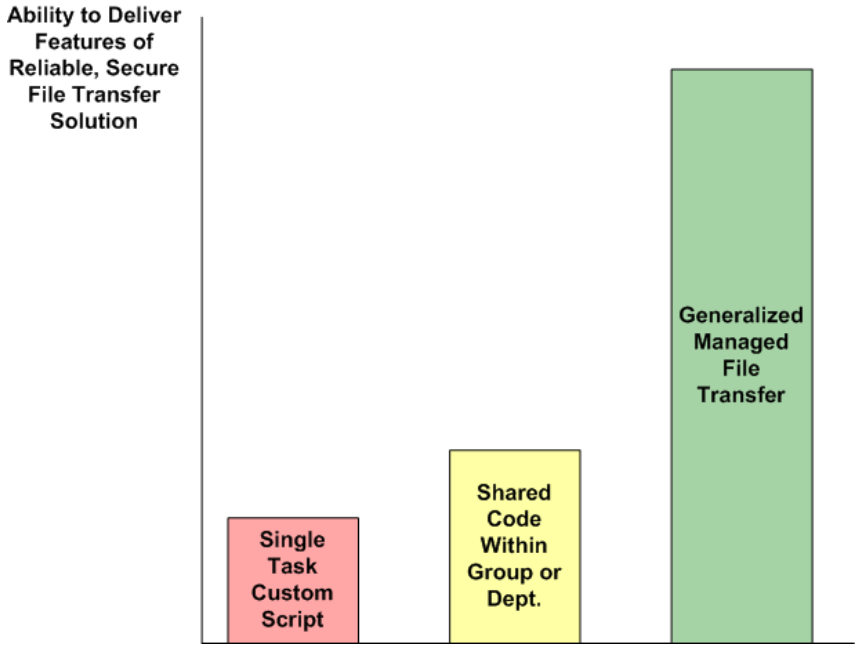

Figure 1.2: Focusing on a single problem limits the range of functionality; taking a more generalized approach for department-level needs is a step toward a general, managed solution.

In some cases, a group or department faces enough file transfer issues that they start to standardize. For example, they may decide to always use Perl for their scripts and they may routinely copy code from old scripts into new ones. These practices lead to common features across scripts, but there are still shortcomings. For example, each script is still a solution to a specific problem and applying that script to a new problem can require significant changes. If someone finds and corrects a bug in one script or adds a new feature to another, those improvements are not available to the other scripts. Sharing a common evolution of code is helpful in some ways, but it is still an insufficient solution for enterprise-scale file transfer needs.

Duplicate Functionality

At the other end of the code sharing spectrum is the opposite problem: duplicating functionality. Given the willingness of developers to share code, post solutions in developer forums, and pass on good resources for new programming ideas, it's safe to assume this is not a group fond of reinventing the wheel. Once again, the pressures to get a solution done and to focus on the bigger picture (usually a project for which file transfer is only a small piece) influences developers to build "just enough" to get the job done.

Why not just go find another piece of code that already solves the problem at hand? The cost of searching out other developers' code, understanding how it works, and adapting it to one's particular needs is often greater than devising your own solution. This is especially the case if the file transfer code includes minimal features and skirts issues such as error handling, verification, and reporting.

There are parallels here to what economists call negative externalities. When a factory emits pollutants into the air, the true costs are not borne by the factory owners; we all share in those costs. When developers write just enough code to satisfy their immediate needs, they risk creating negative externalities for the business at large. Is the code secure? Are passwords stored in plain text in a script? Does the file transfer method meet compliance requirements? Even if you answer 'no' to any of these questions, if the code copies a file from point A to point B, then the code does what it is supposed to do. At least by one set of measures. As you shall see in the next section, unmet business requirements are becoming more prominent criteria in business decision making.

Limited Error Handling

When it comes to error handling, it does not help to be an optimist. In IT, things go wrong on a fairly regular basis. Disks run out of space, programs do not finish running as quickly as we think, and they sometimes return results we don't expect. File transfers are prone to their own set of all-too-common problems:

- Insufficient space to store files
- Insufficient space to decompress files once they are transferred
- Challenges with managing encryption keys for decrypting files
- Protocol-related errors, like dropped packets and lost connections
- Determination of the precise nature of an error instead of simply assuming a copy operation failed

These problems are especially challenging and more likely to occur with large file transfers. They are also challenging to handle well. In fact, developers could easily spend more time writing error handling code than code that actually performs the file transfer functions! It is no wonder, when you build homegrown solutions to limited tasks, error handling often comes up short.

Difficult to Maintain

Custom code can be difficult to maintain. The first version may be clear and logical to the original developers, but as new requirements emerge and other developers add modules and correct bugs, the logic may become less clear and less cohesive. Ironically, the very act of maintaining custom code can decrease the maintainability of the program.

Difficulties arise from two sources. First, there are business requirement difficulties. The original developer may be aware of some non-obvious requirement that she rightly codes for but does not document as well as she should have. Subsequent maintainers may struggle to understand why a particular piece of code is written the way it is or why it is there in the first place. Without knowing the business requirements, another developer may alter the functionality of the program and inadvertently remove an important feature.

The second source is programming difficulty. Sometimes the more efficient the code, the more difficult it is to understand. Code with minimal data structures and tightly written logic may execute quickly but be hard to understand. Scripting languages can pack a good bit of functionality into operators, and programmers can take advantage of side effects of operators to improve speed and reduce the number of lines of code they need to write. This is often desirable but, like so many other aspects of managing file transfers, there are tradeoffs.

There has always been the potential for homegrown solutions to crop up in projects across the enterprise. This problem will likely grow worse with the adoption of public cloud computing. An individual analyst with an idea, some data, and a credit card can take her project to a cloud provider like Amazon EC2, Windows Azure, or any of a variety of other providers. Cloud computing has the potential to help companies develop better insights into their data while cutting their capital expenditures on hardware. Unfortunately, it is also an opportunity to compound existing problems with ad hoc file transfer solutions.

These characteristics of homegrown file transfer solutions are closely tied to a number of shortcomings that are undesirable from a software development perspective as well as from the point of view of business users.

Shortcomings of Homegrown Solutions

The major shortcomings of homegrown solutions can be broadly grouped into five categories:

- Lack of robustness
- Poor adaptability to emerging requirements
- Poor scalability
- Potentially poor security
- Cost of maintenance

These shortcomings are directly related to the adverse business consequence you see with many homegrown solutions.

Lack of Robustness

Robustness is the quality of being able to adapt to a wide range of inputs and operating conditions. Applications that lack robustness, sometimes referred to as *brittle applications*, break easily when explicit and implicit assumptions of the application developer are not met. Imagine a simple but brittle file transfer script that copies a single file from a local directory to a shared network drive:

```
copy c:\temp\data.txt z:\stage\data.txt
```

For such a simple command, it can break in many ways:

- The source file may not exist
- Users running the script may not have read access to the source directory or write access to the target directory
- There may be insufficient space on the target disk
- A file with the same name may already exist in the target directory
- The network mapping specifying the location of the Z: drive may not be defined

These issues do not even touch on the lack of flexibility and adaptability of the script.

A robust script is one that can function with multiple sources and targets, gives the user opportunities to recover from errors, and adapts to non-fatal conditions, such as a file with the same name already existing in the target directory. Certainly, many developers who create custom file transfer scripts have the ability to write robust code; the problem is the time needed to do it. As the list of potential problems shows, there are many ways to break a file transfer application.

With the increasing use of public cloud providers, developers also have to account for file transfers between on-premise servers and cloud storage. As businesses continue to generate, store, and analyze large data sets, there is an increasing need for computing resources for data analysis. Public cloud providers can meet these computing needs, but successful implementation of this type of big data analytics depends upon the ability to transfer data to the cloud reliably and in a timely manner.

Poor Adaptability to Emerging Requirements

Requirements can change in several ways, including the range of inputs that must be handled, the window of time allotted to perform a transfer, the security requirements related to the transfer, and the locations to where files are transferred. Seemingly simple changes can highlight implicit assumptions that do not always hold. For example, the previous simple example assumed we were moving a specific file from one particular directory to another. Granted, this is a trivial example and most of us would have written the script to accept a source file path name and a target file path name. That is just the beginning of the ways you could design the program to be more adaptable. For example, you might want to consider the following questions during the design process:

- Should you assume the copy operation would be between devices on the same network?
- Should you avoid mapped network drives and use server names instead?
- Should you use the syntax specific to Microsoft OS domains (for example, \\servername\shared_directory\) or should you use Internet domain names (for example, datastage.mycompany.com) instead?
- Can you assume there will be a single source and single target files? Should you accommodate the possibility of merging multiple input files into a single target file?
- How will files be transferred from on-premise file systems to object-based cloud storage?

The ability to scale to emerging requirements is another type of potentially new requirement. Scalability is a multifaceted and complex area of file transfer management.

Poor Scalability

Scalability is the ability to continue to function under increased workloads while meeting performance objectives. In the case of file transfer applications, you can increase workloads along at least four different dimensions:

- Volume of transfers
- Frequency of transfers
- Support for new transfer partners
- Integration with new business processes

File transfer applications should be able to scale up, as with the first two dimensions, and to scale out, as with the last pair of dimensions.

Volume of Transfers

The most obvious need for scalability is when the size of files or the number of files increases. In order to meet SLAs, file transfers have to complete in a specified period. Whether it is order information that has to be posted to customer accounts or transaction data that needs to be added to a data warehouse, file transfers have to complete in time for the entire workflow to finish on time. There are several approaches developers can take to improve throughput: compressing files before transfer, using multiple network connections to transfer files in parallel, and using deduplication techniques to transmit duplicated blocks of data only once. These techniques of course add complexity. The effectiveness of each technique will vary depending on the characteristics of the data transferred.

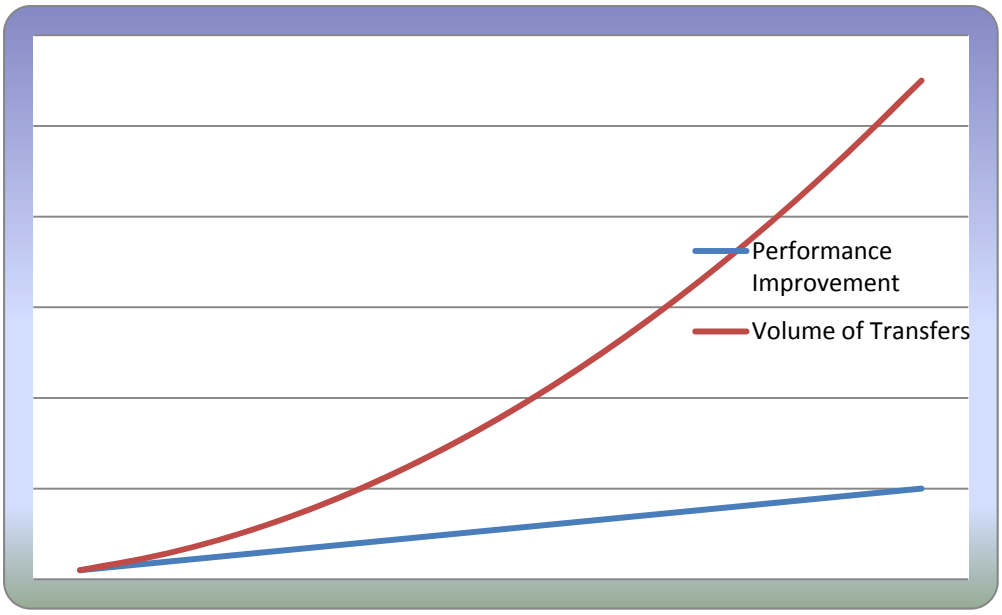

Figure 1.3: The level of performance improvement readily implemented in custom file transfer scripts can be easily outpaced by the growth in volume of transfers.

Frequency of Transfers

Increasing the frequency of transfers can have similar impacts as increasing the volume of transfers. The net effect in both cases is that the amount of data transferred during a given period of time is increasing. In addition to the potential issues outlined with regard to increased volume of transfers, there may be need for additional code to handle problems introduced by more frequent transfers. For example, later stages of processing may not complete fast enough to process previously transferred files. In this case, multiple files will be staged on the target. The transfer program will have to avoid potential filename conflicts when writing files to the staging directory.

Support for New Transfer Partners

Moving on to the scaling out dimensions, let's consider the need to support new transfer partners. Unless a homegrown solution is written in a sufficiently abstract and generalized way, it can be difficult to adapt to new transfer partners, especially with regards to security. Transfer partners may have different authentication mechanisms; how will the homegrown application accommodate those? Will the authentication module have to be rewritten for each new trading partner? How will security information, such as usernames, passwords, or digital certificates, be managed? How will a cloud provider's authentication mechanism, such as authentication keys, be supported? Is there a generic model in place to handle all of these or will there be partner-specific modules? If the answer is the latter, this will significantly curtail the speed with which the transfer program can be used with new trading partners.

Integration with New Business Processes

The last dimension of scalability is the ability to integrate with new applications and processes, which is especially important with the increasing use of cloud computing. In an earlier example, we saw how a BI analyst had to write a custom program to transfer click stream data from a Web server to a statistical analysis package. A file transfer program was already in place for moving data from the Web server to the data warehouse. Ideally, that program should have been used for the click stream data as well. After all, it already worked with the Web server, so issues such as scheduling a process and running it with proper authorizations had already been implemented. A scalable solution is one in which common requirements, like scheduling and authentication, do not inhibit the use of existing transfer applications to new applications. Another common shortcoming in custom, homegrown solutions is they may contain security vulnerabilities that expose the business process to unnecessary risks.

Potentially Poor Security

Information security often entails a balance between security and functionality. The most secure system is also unusable. In the case of file transfers, there are several potential risks that can be introduced with custom solutions:

- Putting usernames and passwords in script in clear text. For example, if ftp is used to transfer files and a username and password is required to authenticate, one could simply code a line such as ftp://username:password@ftp.mycompany.com
- Authenticating with user accounts with more privileges than are required to perform file transfers. For example, using an existing user with elevated privileges rather than creating a dedicated account for file transfers is a potential security risk.

- Using unpatched ftp servers or other applications with known vulnerabilities creates risks. OSs and utility programs used in file transfer should be under patch management controls. This is more difficult if there are multiple copies of vulnerable applications in use.
- If file management practices are not secure, one could compromise the security of the entire process, for example, by failing to minimize the storage of sensitive information in the DMZ of the network and to minimize the number of open ports on the internal network.

As the issue with patching points out, security is dependent on sound maintenance, which brings us to our final shortcoming of homegrown solutions.

Cost of Maintenance

Software maintenance is an ongoing need of any production application. You cannot eliminate the cost of maintenance but you can keep it to a minimum. Many software engineering best practices are designed to make code more understandable, reliable, and robust and that in turn makes the code less costly to maintain. Applying these best practices takes time and a willingness to commit resources to the effort. When you short-change development of file transfer programs because you assume file transfer is an easy task that is only a small part of a larger workflow, you set up your business for increased long-term maintenance costs.

The increased costs are obvious in a couple of maintenance areas. First, there is the cost of troubleshooting. The more dissimilar transfer applications are, the more difficult they are to maintain as a group. There is little or no learning that can be carried over from debugging one application to the next. Transfer programs may be written in different programming languages, use different network protocols, and apply different strategies for improving performance or optimizing for space. The lack of common programming models also makes it difficult to find bottlenecks. The root of a performance problem in one program may be unrelated to the same performance issue in another application. Just as you "reinvent the wheel" each time you create yet another custom file transfer program, you are also creating additional costly maintenance work for yourself or other developers.

The shortcomings commonly seen in homegrown file transfer solutions are not just a problem from a software engineering or IT management perspective; they directly impact business operations outside of IT. The lack of robustness, poor adaptability, poor scalability, potentially poor security, and the cost of maintenance directly contribute to unmet business requirements.

Unmet Business Requirements

Up to this point, we have considered common characteristics of homegrown file transfer solutions and their shortcomings. Many of the shortcomings stem from the fact that many of these programs are designed according to one set of requirements and then you expect them to function against another set of requirements. It is easy to imagine conversations in meetings about new requirements:

- We have a Perl script for transferring files between the Web server and the data warehouse, why don't we just use that to transfer transactions from the customer order system to the data warehouse?
- Let's use the script for downloading data from the West coast parts supplier with that new supplier in Texas.
- Sales volume is up, let's run that file transfer program every hour instead of every night so that we can get more up-to-date figures to the sales managers.

These are all reasonable suggestions on the surface. It is only when you dig deeper that the problems become clear, such as scripts written to run on one OS that won't run on another, transfer partners that use different security systems requiring different application code, and scripts that ran nightly and were not designed to finish in the short timeframes needed for operations during the business day. At this point, you should be starting to see the business impact of homegrown file transfer solutions.

Don't Shoot the Messenger (or in this case, the Developer)

We should note that the problems and shortcomings outlined here do not arise because programmers do a poor job of writing code. Rather, we tend to limit the requirements we give to programmers while at the same time imposing strict timetables for delivering on the application. This results in limited room for the developer to implement more adaptable, scalable, and robust programs. This will only change when we see the need for managed file transfers as instances of a specific type of information management problem and not just an isolated need within a single project.

The unmet business requirements fall into several categories:

- Compliance
- Ability to keep up with increasing need for file transfer services
- Support for multiple uses
- Ease of management
- Cost control
- Support for workflows

IT operations are expected to support compliance with regulations addressing the privacy of personal information and the integrity of business information as well as industry-specific governance issues. Often, to be in compliance, you must not only be doing the right thing, you have to be able to demonstrate that you are doing the right thing. In other words, it is not enough to have a file transfer program that uses encryption to transfer files, you need logs showing those programs were used, the encryption was strong enough (for example, transfer was done with a strong encryption algorithm and sufficiently long keys), and the transfer occurred between two servers with verifiable identities. This level of detailed logging and reporting may not be found in all custom transfer programs.

Business is anything but static. Unfortunately, it is very straightforward to design and implement a file transfer program for today's requirements and maybe a bit more, but it will likely be insufficient for what might be needed in the not too distant future. Project managers who are responsible for implementing a new service or workflow may plan for some increase in scale, but it is unreasonable to expect them to expend time and resources developing a file transfer system that would be useful to another, separately funded project. The logic of project budgeting reduces and even eliminates incentives to support multiple uses and curtails even the willingness to dedicate resources to designing a scalable solution. This is especially the case when scalability issues will only become clear once the system is deployed and maintained out of a separate, IT operations budget.

Ease of management, cost control, and workflow efficiency all fall into a similar category. Project managers, department heads, and developers can all make rational decisions about file transfer and end up with systems that fail with regard to these issues. The real cost of these shortcomings is borne by the business at large. Chapter 2 will delve into more detail about the business consequences of homegrown file transfer solutions. Before turning to that, though, let's briefly discuss a more structured approach to business file transfer.

Defining a More Structured Approach to Business File Transfer

A more structured approach to managing file transfers is required to avoid the shortcomings and adverse consequences of developing and maintaining multiple custom solutions. The first step is to recognize that file transfers are complex tasks that require appropriately designed software. From there, it becomes clear that you cannot develop this level of complex software with the same resources or for the same cost as a custom solution for one isolated use case. Ideally, the structured solution will be a general-use tool that scales up and scales out to meet the demands for file transfer across the enterprise. This requires four distinct functions:

- Centralized management for defining transfer jobs, monitoring the status of executing and scheduled jobs, prioritizing tasks, and other administrative functions
- Security services to manage authentication with transfer partner systems, encrypt confidential data, log transaction details, and enforce security policies governing file transfers; security services should work with reporting and logging services to keep administrators informed of significant security events, such as failed login attempts during transfers or instances of insufficient privileges to perform a transfer

- Task specification mechanism to define transfers—this should include the ability to define the schedule of transfers, choose error handling options, define security requirements, securely capture and store usernames and passwords, and specify triggers or event processing steps, such as executing a script upon successful transfer. The task specification mechanism should support transfers to business partners, including cloud providers.
- Reporting and logging services are needed for both tracking individual transfer jobs and for monitoring global trends, such as increases in the number of files transferred, the volume of data transferred, error rates, and security-related issues

Each of these services compensates for some of the shortcomings commonly seen in custom solutions. Although it is not listed, scalability is a critical feature of a managed file transfer framework. Unlike the reporting services or task specification mechanism, scalability is a global property of the way the system is designed. Scalable systems will use a combination of features, such as compression protocols, multiple connections, and multi-threaded processes, to avoid bottlenecks and maintain sufficient throughput.

Scalability is a key consideration when supporting large-scale analysis operations, commonly referred to as "big data." Big data typically includes analyzing large data and sometimes diverse data sets, such as various types of logs. These data sets are generated and stored on many servers but have to be consolidated in a cluster or cloud computing environment for analysis. In order to keep pace with these volumes of data and the pace at which they are created, it is imperative that you have a scalable file transfer solution in place.

The chapters that follow will further examine how a structured framework for file transfer works to address the unmet business needs that remain when custom solutions are in use.

Summary

The need to transfer files is ubiquitous in IT operations. We are constantly moving data between desktops and servers and between servers within a business, but we are moving data between transfer partners as well, including cloud computing providers. It is easy to see each instance of the need for a file transfer as just another task in a more complex workflow. When you do that, you tend to create custom homegrown solutions to solve the immediate problem at hand. Over time, as you keep repeating this pattern in other projects and in other parts of the organization, you will find that you have created a sprawl of file transfer programs with some unfortunate shared characteristics. The most important are lack of scalability, costly maintenance requirements, poor security, and an inability to adapt to changing business requirements. These, in turn, lead to a host of unmet business requirements. Fortunately, by considering these file transfer needs as instances of a more general IT operation, you can justify dedicating resources to solve the enterprise-scale problem facing the business. The forthcoming chapters will examine how managed file transfer solutions address enterprise requirements for secure, reliable, and robust file transfer services.