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 3: Selecting a File Transfer Solution: 7 Essential Requirements

No one intentionally sets out to deploy an insecure and unreliable file transfer system. There are no business drivers that prompt IT professionals to look for a system riddled with security risks or likely to fail during the course of normal operations. Problems with security and reliability often creep in unnoticed when developers focus more on delivering a finished product on time and on budget. This is especially a problem when file transfer is seen as a minor requirement or a simple task that does not warrant as much attention and consideration as other aspects of system design. This is a myth.

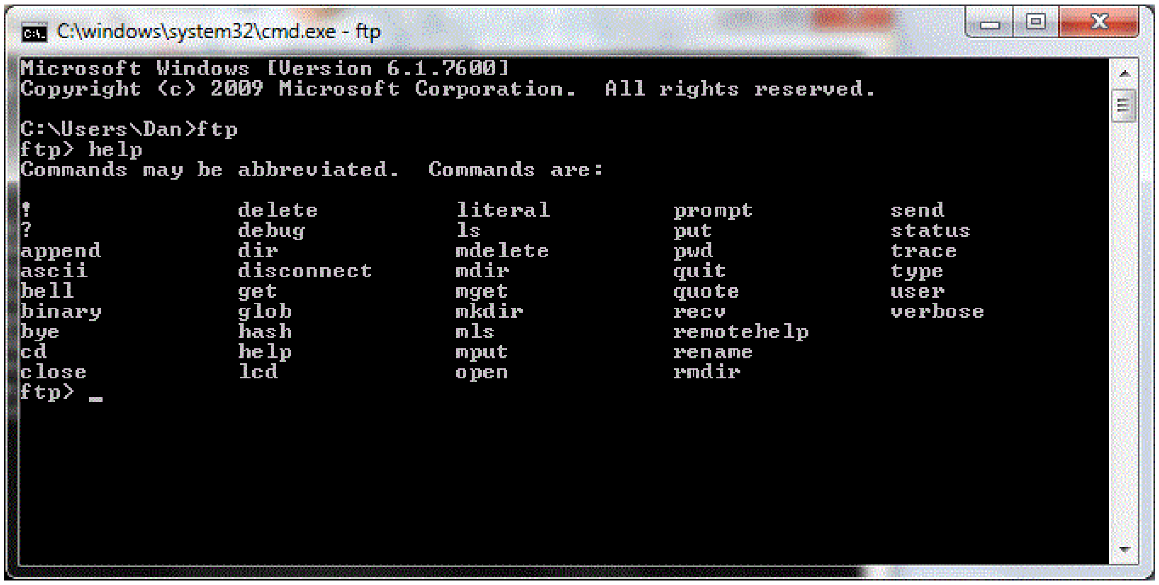
As we have seen in the previous two chapters, enterprise file transfer is a complex process with multiple sources of requirements, a wide range of constraints, and a target for security threats. The purpose of this chapter is to identify essential requirements for a secure and reliable file transfer solution. This identification should help frame discussions about design around actual requirements instead of myths that can thwart your long-term objectives. Before examining seven essential requirements of a secure and reliable file transfer system, let's dispel misunderstandings that can cloud your understanding of file transfer.

Dispelling a Few Misunderstandings About File Transfer

The first two chapters of this guide dispelled common myths about file transfer: that it's easy, should not require much code, and is basically a glorified file copy operation. Myths and misconceptions such as these stem from three fundamental misunderstandings.

File Transfer Is a Simple Technical Operation

Misconceptions stem from a narrow focus on technical aspects of implementing file transfer. For example, the file transfer protocol (FTP) was obviously created to transfer files from one system to another. FTP includes simple commands, such as "put," which copies a file from the system running the FTP command to a remote system, and "mget," which copies multiple files from a remote system to the system running the FTP command. By thinking of file transfer in terms of implementation and simple protocols, you can easily fall into the trap of thinking design and development efforts should focus on determining which FTP commands to use and how to wrap them in a Perl script or any of the other popular programming languages.



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Dan>ftp
ftp> help
Commands may be abbreviated.  Commands are:

!               delete          literal         prompt          send
?               debug           ls              put             status
append          dir             mdelete         pwd             trace
ascii           disconnect      mdir            quit            type
bell            get             mget            quote           user
binary          glob            mkdir           recv            verbose
bye             hash            mls             remotehelp
cd              help            mput            rename
close           lcd             open            rmdir
ftp> _
```

Figure 3.1: How complicated can file transfer be if a small set of commands can implement FTP? This is not a good indicator of the complex requirements of file transfer.

There Are Few Risks, Potential Errors, or Changes to Requirements

Second, many fail to account for the potential problems with file transfers. Perhaps designers and developers are inherently optimistic about the problems you work on and the applications you develop. This optimism is unwarranted with file transfers. The litany of potential problems includes:

- Insufficient storage space on the target server
- Access control restrictions that prevent successful transfer
- High-latency networks that slow the transfer to unacceptable levels
- Security risks such as file tampering and information theft
- Change in requirements that necessitate modifications to overly specialized code

These problems are obvious, especially when they disrupt file transfer operations. Less obvious are problems that stem from poorly integrated file transfer processes.

File Transfer Is an Isolated Operation

This is the most ironic of all misunderstandings about file transfer. After all, why bother transferring files if they were not needed for some reason? Perhaps this is not so much a misunderstanding as an under appreciation. Consider, once a file is transferred, what happens to it next? Some things you need to consider at this point are:

- Is the file sufficiently secured in the directory to which it was transferred?
- Could another user accidently overwrite this file before it is copied, processed, or otherwise operated on?
- What process will remove the file from the directory when it is no longer needed?
- How will other processes be notified that a file has arrived and the next stages of processing should be initiated?

No file is an island unto itself. Once it is transferred, there are likely other operations that have to be performed. Even when post-transfer requirements are minimal, there will likely be some requirements for post-processing. For example, when a file is transferred for backup purposes, the event should be logged for management purposes.

The 7 Essential Requirements of Secure and Reliable File Transfer

To transfer files in a secure and reliable manner requires more than may be apparent at first. Once common misconceptions are dispelled, you can see that common file transfer requirements span organizations of all sizes. Seven of the most important are:

- Support for both internal and external information flows
- Support for multiple protocols
- File transfer security
- File transfer optimization
- Reliability and transaction control
- Alerts and process reporting
- Event processing and automation

These requirements address the way file transfer operations are used in today's businesses. They start with the business driver perspective—not a technical orientation that drives the myths we saw earlier. These requirements can be contrasted with those myths and misconceptions to see how the reality of business operations is served by these requirements and not by common misconceptions.

Myth	Reality	Relevant Requirement
File transfer is a simple technical operation	There is more to file transfer than copying bytes. Business requirements dictate the need for performance, security, and reliability across a range of applications—many with varying technical requirements.	Support for internal and external information flows Support for multiple protocols
There are few risks, potential errors, or changes to requirements	There are many ways a file transfer operation can fail. Private and sensitive information may be at risk; security controls are required.	Security File transfer optimization Reliability and transaction control
File transfer is an isolated operation	Files are transferred for a reason. Other processes may need to process the data transferred or log the fact that it was transferred.	Alerts and process reporting Event processing and automation.

Table 3.1: Common myths and misconceptions undermine business objectives with regards to file transfer. The seven essential requirement outlined here are meant to support those business objectives.

Essential Requirement 1: Support for Internal and External Information Flows

Internal and external information flows are different enough that you need to select file transfer solutions based on their ability to support both. The differences stem from the fact that with internal transfers, you are typically, but not always, working with shared identity management systems, common access controls, and transmissions over secure routes. Support for both internal and external information flows means there is support for:

- Common requirements
- Multiple access controls
- Encrypted transmissions

Common Requirements

The common requirements for internal and external information flows include:

- Sufficient performance
- Adequate logging of events in the transfer process
- Ability to schedule jobs
- Ability to recover from errors and restart transfer jobs

Sufficient Performance

Performance is a fundamental requirement for any IT service; file transfer is no different. The specific level of performance will vary by application, but in many cases, it is determined by windows of time in which files can be transferred and the number and volume of files that must be transferred in that time.

Adequate Logging of Events in the Transfer Process

File transfers can entail multiple events. Many of these are expected, such as initiation and termination of connections as well as the start and completion of a transfer. Others events are not unexpected but they are unwelcome. These include errors, such as insufficient storage space or overly restrictive access controls. They also include reports of failed services, such as a lost network connection that prematurely terminates a file transfer. All of these should be logged. Log information is especially useful for diagnosing problems with individual transfer operations, but they can also provide information on trends, such as increasing time required to complete transfers or increasing frequency of dropped packets. This type of information can be useful to track down emerging problems or plan for future capacity requirements.

Ability to Schedule Jobs

We often repeat the file transfers over and over again. Copy the database export to a report server that is used to off load reporting from the production server. Transfer backup files to the disaster recovery site. Upload data from branch offices to headquarters. Send business partners nightly updates to the product catalog. As business processes span departments within an organization as well as across organizational boundaries to business partners, there is the potential for more file transfers. To ensure these transfers occur as needed, a file transfers solution should provide for scheduling these tasks.

Ability to Recover from Errors and Restart Transfer Jobs

One thing about file transfers is virtually certain: If you do enough of them, you will eventually run into errors. This is true for both internal and external transfers. Of course, how you deal with them can be very different. For example, when an external server runs out of space, you cannot always submit a service desk ticket to get more space. (That may be true of internal processes as well). Once the problem is resolved, the transfer process should resume with minimal intervention. The file transfer system may be configured to attempt the transfer again at regular intervals, ideally starting for the point of failure and not restarting from the beginning of the file. Each attempt should be logged as well; the number of attempts and time required to resolve the problem is useful information.

Support for Multiple Access Controls

When transferring files across organizational boundaries, you have to support multiple forms of access controls, including different types of authentication. Many organizations can meet their security requirements using username and passwords to authenticate file transfer users. Others may have stricter controls and require that all users authenticating to a file transfer service use a digital certificate-based method for authentications.

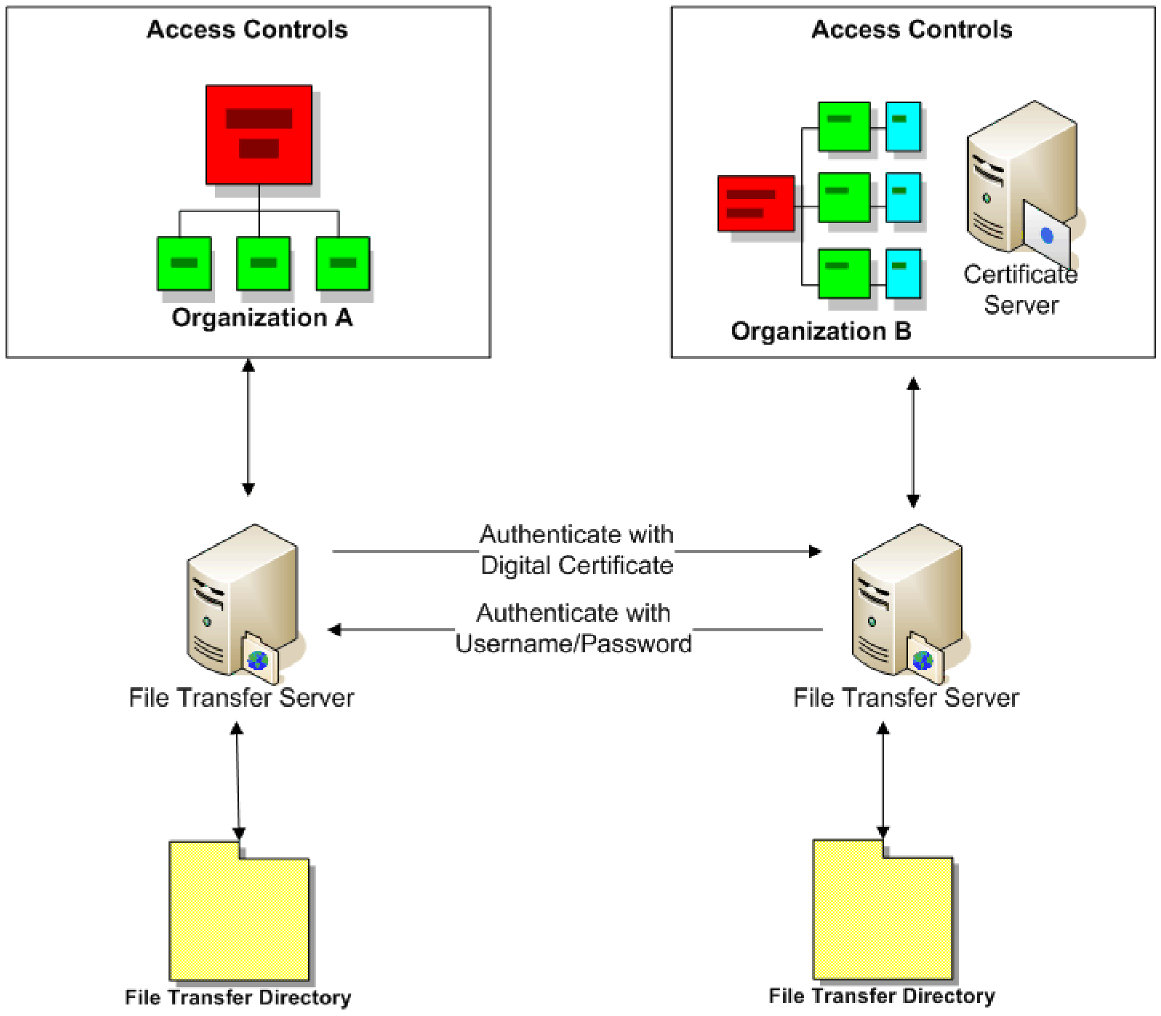


Figure 3.2: Different organizations may use different methods for authentication, such as username and passwords or digital certificates.

Support for Encrypted Transmission

Transferring files across the public Internet exposes them to threats such as tampering and divulging confidential information. To mitigate the risk from these threats, file transfers should be encrypted. There are several ways this can occur, although they all rely on the same basic encryption technologies:

- Encrypting each file individually as part of the file transfer process
- Using an encrypted file transfer protocols, such as SFTP, which encrypts all traffic related to the file transfer
- Using a virtual private network (VPN), which encrypts all traffic, including file transfers

When evaluating encryption applications, make sure the applications are using sufficiently strong encryption. Encryption strength is something of a moving target. An old encryption standard, DES, was once considered state of the art; today, DES encryption could be easily broken. Look for encryption programs that are Federal Information Processing Standard (FIPS) verified; use strong encryption algorithms, such as the Advanced Encryption Standard (AES); and employ sufficiently long encryption keys.

At this point, we are starting to move from the high-level requirement of supporting both internal and external information flows to discussing lower-level implementation details. This brings us to the second essential requirement, the ability to support multiple protocols.

Essential Requirement 2: Support for Multiple Protocols

There is an old saying in the Perl community, "There is more than one way to do it." The sentiment reflects the fact that there are often many ways to use the tools and techniques available to us to accomplish a specific task. That sentiment applies equally well to file transfers. Take, for example, the number of protocols available for file transfer. There are a number of open standards as well as proprietary protocols that may offer added benefits, such as accelerated file transfer.

The flip side of having many protocols to choose from is that your transfer partners may use any one or more of these protocols. That, in turn, requires you to support what your trading partners support. Let's look at the variety of protocols and the advantages and disadvantages they present.

Open Standard Protocols

As a general rule, open standards mean greater interoperability than proprietary standards. When it comes to file transfers, there are several protocols to choose from; some are relatively old (at least relative to the age of the Internet) and others are more recent advances with improved features.

The most important protocols for file transfer include:

- FTP—Dating from 1971, this is the earliest file transfer protocol that provided a basic client/server model for file exchange. The protocol has been updated since then with the latest version defined in RFC 959 adopted in 1985.
- SFTP—Secure File Transfer Protocol brings basic file operations to the Secure Shell Protocol (SSH). The underlying SSH protocol provides security features, such as authentication and encryption.
- FTPS—This protocol builds on FTP by adding security features to the protocol with the use of Secure Sockets Layer (SSL) and Transport Layer Security (TLS).
- HTTP—The Hypertext Transfer Protocol is best known for transferring Web page content but has been used for basic file transfer.
- HTTPS—HTTP over SSL provides the same basic functionality as HTTP but adds the security features of SSL, such as encryption, as well.
- AS2—Formerly known as Applicability Statement 2, the AS2 protocol builds on the secure mail protocol S/MIME to incorporate encryption and digital signing to files that are transferred as “attachments.” AS2 also provides for message disposition notifications, which can indicate the status of the message and file after reaching the recipient.

It is worth noting that file transfer protocols are adapting to meet the essential requirements of complex business operations by incorporating security and notification services. They do this, in part, by employing file transfer protocols with other existing protocols.

Proprietary Protocols

FTP is designed to use TCP for transmission. TCP is a general purpose transmission protocol that ensures reliable, in-order delivery of data packets. Web pages and emails are sent over TCP as well. TCP is a general solution that can meet a wide range of requirements. A drawback of TCP, though, is that the exchange of control information between a client and a server can impose substantial overhead on a transmission.

Some managed file transfer vendors implement proprietary techniques to improve performance of file transfers. For example, a vendor may use UDP and implement additional functionality on the client and server to ensure reliable, in-sequence transfer. As with any proprietary solution, the client and server will have to agree on use of this protocol and possibly install proprietary software. Another potential advantage is that vendors may incorporate other features, such as compression, that can further improve overall performance.

Anyone who has lived through complex application development and deployment efforts that have been hampered by proprietary protocols may have something of a knee-jerk reaction against proprietary protocols. For example, a proprietary protocol within a company may function without a problem because all parties are using the same protocol; however, when you try to use the same protocol with a business partner that does not support that protocol, problems arise.

You just need to remember that open protocols are valued for the way they allow you to meet business requirements; there is nothing inherently valuable about an open protocol. It is a means to an end. Similarly, there is nothing inherently wrong with a proprietary protocol; but it will have to prove itself. That is, whatever the proprietary protocol has to offer must outweigh the lost flexibility, interoperability, and other benefits of open standards. These benefits can be in the form of accelerated file transfer, improved recovery after an error in transmission, greater control over the transfer process, or some other improvement to the overall process.

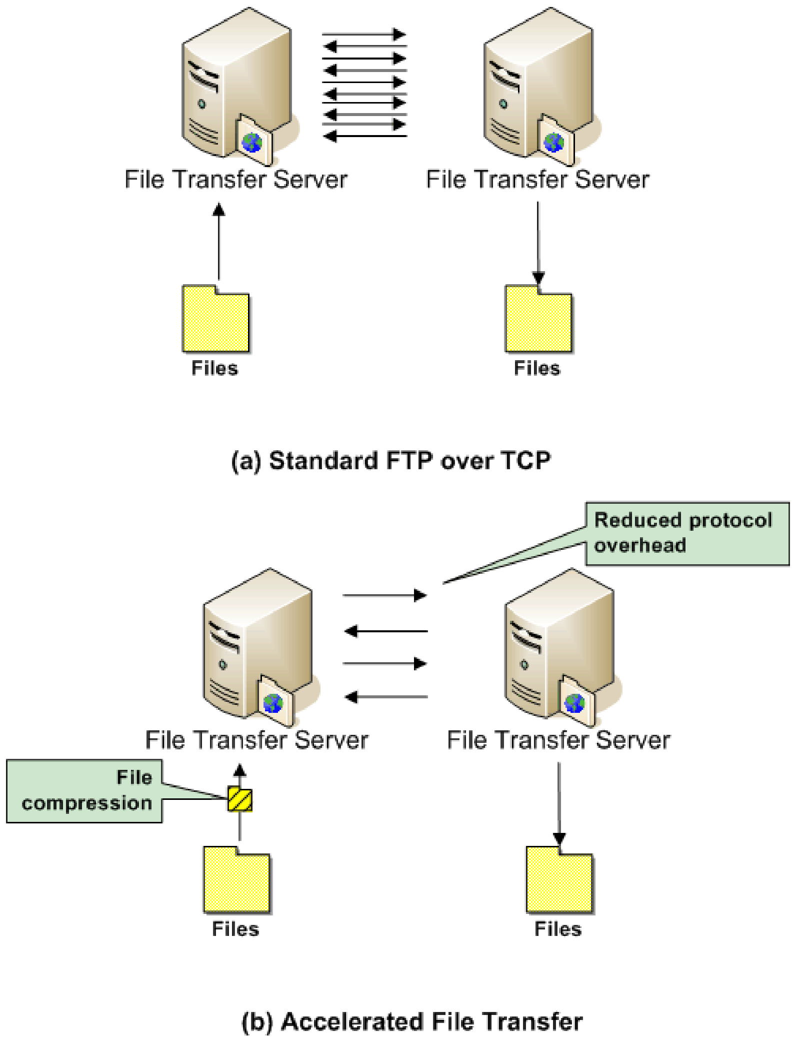


Figure 3.3: Proprietary protocols can accelerate file transfer through a combination of compression and reduced overhead in the transmission protocol.

Essential Requirement 3: File Transfer Security

Data managed by applications, stored in database, and stored in user directories can be protected with a variety of access controls. When data is in transit from one system to another, there are risks. In particular, confidential or other protected data is at risk if it is staged in the DMZ and if it is transmitted in unencrypted form.

Staging Data for Transfer

Staging areas are shared resources where you temporarily store files so that they can be transferred to the final target server. By definition, multiple users put files into the staging area. This typically implies fairly open access controls; at the very least, many different users can write to the staging area.

In some cases, you might be able to treat the staging area as accessed by a trap door: you can write data but cannot read it. Data goes in one way but cannot be accessed in the same way. This helps reduce the risk of another staging area user reading your files, but this setup creates other difficulties. For example, without read access, you cannot readily confirm the file was written correctly by calculating a checksum on the written file. You do have other methods for protecting confidential data, though.

Confidential information should be encrypted before it is copied to a shared staging area. In addition, files should be deleted from the staging area after they are transferred to their target locations. In the event that a file is not successfully transferred after some reasonable period of time or number of attempts, the file should be purged from the staging area. This will prevent a file from remaining in a staging area for extended periods of time because of errors or problems with the target server. This is especially important when the target server is controlled by another party that might not have the same level of concern about your confidential information sitting in a shared staging area.

File Transfer Security in the DMZ

The DMZ part of a network is semi-trusted. The DMZ is protected by a firewall, which offers basic security controls such as blocking ports to unused services. At one time, that actually provided a fair degree of protection. Today, however, such is not the case due to the amount of traffic that moves into and out of the DMZ over HTTP, including security threats. In general, it is not reasonable to assume confidential data is protected in the DMZ. When compliance is a concern, confidential and proprietary data should not be stored in the DMZ. In fact, a good rule of thumb is to not store data for file transfer in the DMZ, period.

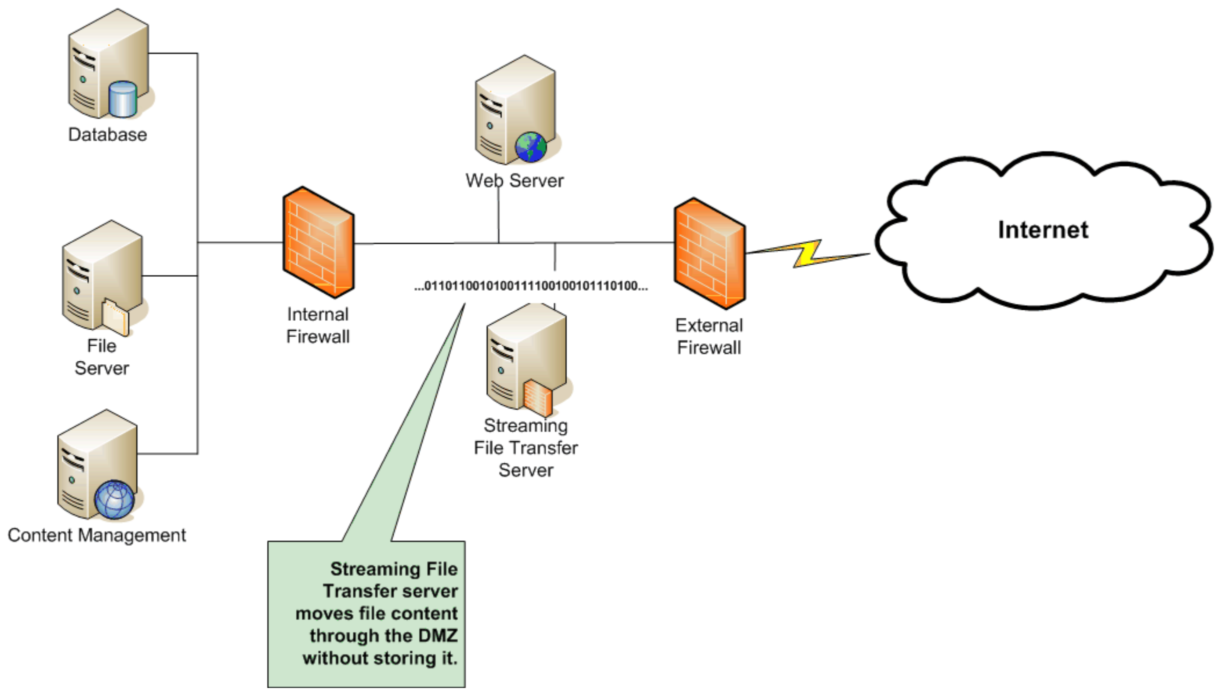


Figure 3.4: Files are vulnerable to compromise if stored in the DMZ. File streaming moves file data immediately through the DMZ without exposing it to risks associated with persistent storage in the DMZ.

The alternative to streaming is a store and forward model. Consider a simple example. An insurance company and a hospital have established a procedure for transferring patient billing information on a daily basis. The hospital transfers patient and procedure information to the insurance company, and the insurance company provides coverage and payment information in return. The hospital has a DMZ and an internal firewall that blocks any direct access to the trusted network by any outsider, including business partners. To enable file sharing, the hospital IT department decides to set up an FTP server in the DMZ. Outsiders, such as the insurance company, can access the ftp server, pick up patient and procedure information, and leave payment and coverage information. In an ideal world, this may work, but not in our world.

This store and forward approach leaves confidential data vulnerable to a number of security weaknesses with FTP. By leaving data in the DMZ, any attacker with the ability to tunnel threats over HTTP can get into the DMZ, access the Web server, and then exploit vulnerabilities on that server to gain access to other systems in the DMZ, possibly the file transfer server. Streaming data through the DMZ does not sacrifice the advantages of having a DMZ, but it avoids the risks associated with storing data there as well.

File Transfer to the Cloud

Data transfers to the cloud also require attention to security concerns. The level of security provided by your network can vary by the way you use the cloud. Let's consider a couple of different scenarios.

In one scenario, you use public clouds on occasions when you need a large number of servers to analyze or process a large amount of data. This situation can take place when you are doing a market analysis study and you need to run data mining and statistical analysis programs on years of customer transaction data and third-party demographic data. For occasional uses like this, you might transfer data from your on-premise servers to cloud storage. You are responsible for securing communications between your on-premise servers and the cloud. You should ensure authentication information is protected and that confidential and private data is encrypted before it is transferred.

In the second scenario, your company uses cloud servers almost continuously. You treat the cloud as an extension of your internal resources. In this case, you might implement a virtual private network (VPN) between your corporate network and the cloud provider's network. File transfers are encrypted by the VPN software, and application-specific encryption is not needed.

A well-designed file transfer solution should accommodate either scenario; custom scripts that assume a secure network might not adequately protect confidential data when used in the first scenario.

Essential Requirement 4: File Transfer Optimization

As file sizes and volumes of transfers grow, it is imperative that a file transfer solution be able to optimize transfer operations. Optimization in this context can be thought of in terms of automation and efficiency.

File Transfer Automation

File transfers should require minimal manual intervention. This automation is achieved through a combination of robust scripting and scheduling features and policy definitions governing the transfer process.

A robust scripting environment is required in file transfer operations. (This is probably one of the reasons Perl is such a popular choice for ad hoc file transfer programs). Scripts allow you to collect the files needed in a transfer set, perform pre-processing operations, and respond to events in a file transfer workflow. In cases where there are frequently required operations, such as transfer all files in a directory or transferring all files with names that match a regular expression, the file transfer system may provide that as a standard operation.

Scripts can become complex when automating file transfers to the cloud. You pay for cloud computing resources based on the time you run virtual machine instances, so you will want to minimize the time your virtual machines are waiting for files. You might, for example, script your file transfers in a way that triggers another script to start virtual machines in the cloud only after the transfer is complete. Ideally, a file transfer solution will support workflows that allow for scripts to perform tasks in additional to transferring files.

Policies are templates for operations and workflows that can be applied in multiple instances. For example, the sales department may need to upload daily sales data to the data warehouse every night. The data is actually distributed over a number of geographically disperse servers, but they all use the same file structures to store the data intended for the data warehouse. A policy could be defined to transfer files based on the current date found on the files in the staging directory on each server. The policy, which may be implemented as a script or a configuration file executed by a script, may include commands to trigger other events once the file transfer is complete, such as starting a load process in the data warehouse.

Automation reduces the need for manual support in file transfer operations. This opens the possibility of performing more and more file transfers. That is, of course, unless the file transfers system cannot keep up the volume of data.

Efficiency of File Transfers

The efficiency of a file transfer is a measure of how much data can be transferred in a given amount of time. One obvious way to improve this metric is to increase the bandwidth of the network and improve the I/O throughput of servers; however, we are more concerned with the file transfer software and processes than the infrastructure they run on. That leaves two options to improve throughput of file transfers: reduce the amount of data transferred and reduce the overhead associated with each transfer.

Reducing volumes can be done with compression. Standard compression algorithms for data transfers do not lose information but reduce the size of the representation required to encode that information.

Reducing the overhead associated with the transfer can be done with accelerated file transfer protocols. As noted earlier, these proprietary protocols use different techniques than those used in TCP to transfer data while still ensuring delivery of all data in the proper order.

Accelerated file transfer protocols are especially important to transfer large files to the cloud or to business partners. Big data analytics is a promising evolution of business intelligence practices; however, it can require significant computing resources. Public cloud providers can readily offer the computing resources needed, but you have to get your massive data sets to the cloud first. Optimized file transfer can help reduce a barrier to entry to big data analytics by making public cloud providers a viable option for computing resources.

Essential Requirement 5: Reliability and Transaction Control

Additional key requirements for file transfer solutions are reliability and transaction control. The two are closely related, so we will consider them together.

Reliability pertains to the ability to transfer files consistently over time. It also implies a degree of robustness; simple errors should not completely disrupt or disable the transfer process. A reliable file transfer system will be able to handle a range of operating conditions and requirements:

- Transferring large files
- Transferring a large number of files
- Completing transfers under adverse network conditions, such as high latency
- Recovering from disrupting errors, such as lost connections

Some of these requirements are met in part by other features, such as accelerated file transfer that contributes to transferring large files and operating on high-latency networks. Others, such as recovering from disruptive errors, require additional functionality.

One way to implement a recovery mechanism is to use checkpoints. Checkpoints are control mechanism used to ensure the transfer process up to a point has been successful. Metadata about the transfer process and the state of the transfer is also recorded. In the event of an error, the file transfer system will not have to start the transfer all over again from the beginning; instead, the process can restart from the last checkpoint. Checkpoints are essentially the last known good state of the transfer.

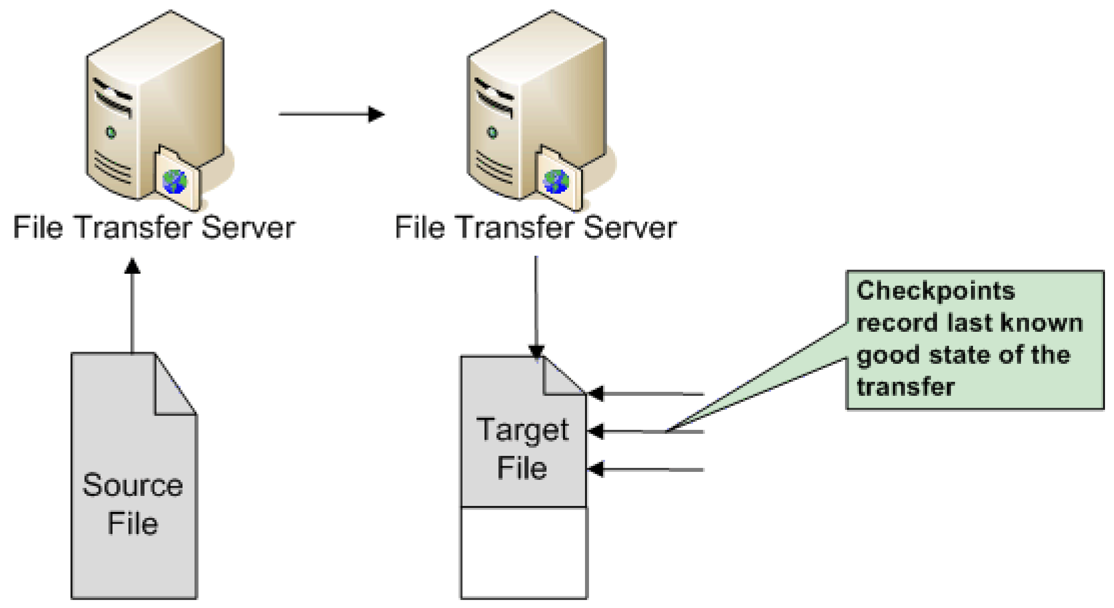


Figure 3.5: Checkpoints are metadata structures that record information about the file transfer process that allow a disrupted transfer to resume from the latest checkpoint rather than starting from the beginning of the file.

Essential Requirement 6: Alerts and Process Reporting

It is not enough to reliably and securely transfer files between systems, you must be able to manage the process as well. This brings us to the sixth essential requirement: the need for alerts and process reporting. For the purposes of this discussion, we distinguish alerts, which are notifications of some unexpected event, from process reports, which are reports about the status of normal operations and activities.

File Transfer Alerts

We have discussed in some detail the various ways things can go wrong with file transfers and the need to handle these situations. One of the elements of effective error management is reporting when a problem occurs. A problem report should include enough information for a systems administrator to understand which file transfer is involved and what the technical aspects of the problem are (at least to some degree). This is accomplished by including information such as the following in error reports:

- Name of the job that failed
- Severity level
- Name and size of the file that was in the process of transferring when the failure occurred
- The source and target destinations of the file transfer
- Usernames of authenticated users on both sides of the transfer
- Summary of error message returned by subsystem that detected the failure, such as the operating system (OS), Java runtime environment, and so on

Note that alerts do not have to contain all relevant information about an error, just enough to make the systems administrator aware of the problem, its severity, and a basic description. Alerts can be delivered on a number of platforms, such as email and text messages, so you must be careful not to overload an administrator's cell phone with a message too long to read on a mid-range mobile device. Details of the failure are best tracked in log files that can be analyzed in more detail as needed.

Process Reporting

Alerts tend to be bad news; process reporting is where you get the good news, and occasionally some bad news. Process reporting provides details to administrators so that they can (1) ensure operations are completing as expected on a day-to-day basis and (2) track trends over longer periods. The level of detail can vary in process reports. In some cases, a simple message that a transfer is complete is enough. In other cases, administrators will want detailed reports about all transfers over a given period of time. These may include details such as:

- The number of transfers in and out
- Total volume of data transferred in and out
- Time required to complete transfers
- Breakdown of transfers by time, department, transfer partner, and so on
- Number of errors by severity level
- Number of restarts due to errors in transfer
- Percentage of successful restarts

Process reporting is essential for any business operation and file transfers are no different.

Essential Requirement 7: Event Processing

The last of the essential requirements we consider is event processing. Transferring a file from one system to another is often a multi-step process. Furthermore, it is not uncommon for a single file transfer job to include multiple files. You can roughly divide event processing into two types: pre-processing activity and post-processing activity.

Pre-processing activities are those required to prepare files for transfer. These activities include tasks such as:

- Collecting files and copying them to a staging area
- Verifying the structure and content of files
- Applying basic transformations on files, such as reformatting to meet the needs of the target system
- Logging details about file preparations
- Encrypting files prior to transfer
- Compressing files prior to transfer

Post-processing events typically trigger tasks that make use of the recently transferred files and include:

- Triggering the next processes in the workflows, such as importing files into application databases
- Logging data about the size and number of files transferred
- Copying files from staging areas to long-term storage directories with appropriate access controls
- Decrypting files
- Uncompressing files
- Generating process reports

Pre- and post-processing operations can be complex, especially when dealing with data imports and exports that require extensive data quality checks, error reporting, or trigger processing and analysis workflows in the cloud. A file transfer solution should include sufficient support for process automation to meet the pre- and post-processing needs of the organization.

Summary

File transfers are used many ways, but the essential requirements outlined here are broadly applicable. To summarize, the seven essential requirements are:

- Support for both internal and external information flows
- Support for multiple protocols
- File transfer security
- File transfer optimization
- Reliability and transaction control
- Alerts and process reporting
- Event processing and automation

These requirements grew out of the way file transfers are typically done in businesses and other organizations. File transfers are common operations in more elaborate workflows and involve sensitive and confidential data that must be protected. The file transfer processes themselves are often part of mission-critical operations and therefore must be secure, reliable, scalable, and manageable. The seven essential requirements translate these high-level requirements into functional requirements that should be available in enterprise-quality file transfer solutions.