SDS VITALSIGNS

VitalSigns for FTP™

VFTP

Managed FTP for z/OS

Automation • Auditing • Security • Monitoring

Transforming standard z/OS FTP
into a true mainframe-caliber utility

Software Diversified Services

CONTENTS

QUALITY MAINFRAME SOFTWARE SINCE 1982

Software Diversified Services delivers comprehensive, affordable mainframe and distributed software with a focus on cybersecurity and compliance. Hundreds of organizations worldwide, including many Fortune 500 companies, rely on SDS software. Our expert development and award-winning technical support teams are based in Minneapolis, MN. To learn more, please visit *www.sdsusa.com*.

Software Diversified Services
1322 81st Ave. NE
Minneapolis, MN 55432
(763) 571-9000

Managed FTP for z/OS
Automation · Auditing · Security · Monitoring

File Transfer Protocol (FTP) usage in z/OS environments continues to grow, incessantly, both inbound and outbound—with many of these transfers still being unautomated, unregulated, unsecured, and unmonitored. Since it is not unusual for a mainframe to participate in tens of thousands or even a hundred thousand transfers a day, using FTP unmanaged cannot and should not be condoned. Unmanaged FTP violates the basic tenets of mainframe operations, compromises the integrity of mission-critical data, unnecessarily depletes mainframe MIPS, and undermines enterprise security.

Hackers around the world are intimately familiar with all of the foibles of standard FTP. Using standard z/OS FTP without an additional layer of automated, real-time operational management creates a glaring, significant point of vulnerability that is easy to exploit.

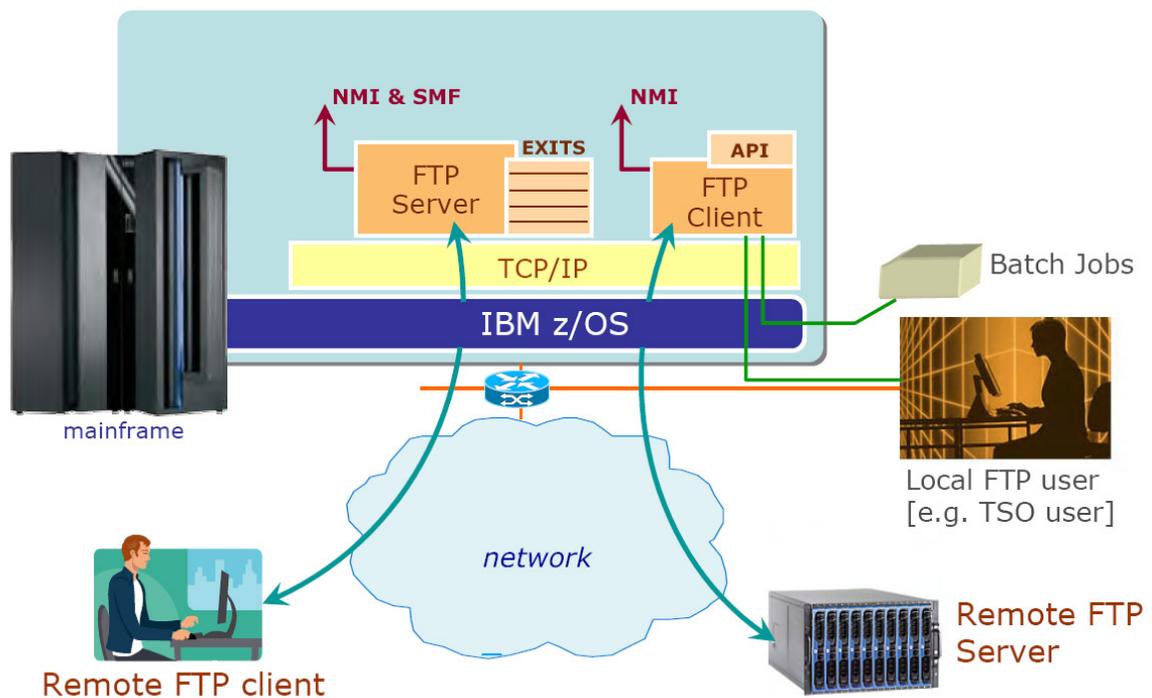Unmanaged FTP is an easy-to-find, unlocked, unguarded back door into your mainframe.

The lack of adequate FTP automation with programmatic error-handling and retry capabilities, furthermore, needlessly confounds z/OS batch processing, disrupts operational schedules, tests the resolve of help desk personnel, jeopardizes compliance, and in general saps both user and enterprise productivity. Suffice to say that standard FTP, unmanaged, is unsuitable for use in today's mainframe environments. Of that there should be no doubt or debate.

Z/OS FTP: LIKE ACHILLES, A WEAK HEEL UNDERMINES OVERALL STRENGTH

z/OS, via its Communications Server suite, provides full-fledged, standards-compliant FTP capability with multi-IP-stack support that is robust and scalable. This FTP functionality consists of a z/OS FTP server and a z/OS FTP client. The FTP server handles FTP requests from remote clients (e.g. downstream PCs or distributed Unix systems) while the FTP client enables mainframe endpoints, whether batch jobs or real-time terminal users (e.g. TSO users), to interact with remote FTP servers.

The server and client both support SSL/TLS-based security and generate Network Management Interface (NMI) records for certain key events. The server includes callable program exit processing. The server can also produce System Management Facility (SMF) records that duplicate the data contained in the NMI records.

The FTP client contains an API which can be used to programmatically drive the client as well as to minutely monitor its operations—on a command-by-command basis, if required. Appropriate ancillary software is, however, needed to take advantage of this FTP client API. There is no equivalent API currently available for the z/OS FTP server—though, with ingenuity, the server's post-processing exit routines may be used to extract some comparable operations-monitoring data. Again, this requires third-party software.

Basic makeup of z/OS FTP.

The low-level and somewhat ad hoc management-related features of z/OS FTP cannot be construed as offering sufficient security, automation, or real-time and historical monitoring for mission-critical, high-volume mainframe FTP operations. That is irrefutable.

To have the necessary automation, monitoring, and management, z/OS customers have no choice but to implement an ancillary FTP manager. Such an FTP manager, if well architected, will gainfully exploit, synthesize, and augment the IBM-provided FTP-management hooks and stubs, such as NMI, the client API, and server exits, to ensure that z/OS FTP can truly qualify as a mainframe-class, managed service. VitalSigns for FTP (VFTP), the focus of this white paper, is a good example of a well-architected, state-of-the-art z/OS FTP manager. VFTP, as will be shown, adroitly addresses FTP automation, security, auditing, and monitoring.

The bottom line here is that standard z/OS FTP is robust and scalable, but woefully inadequate management is its undeniable Achilles' heel. Hence the immediate justifiable need for a product such as VFTP to transform z/OS FTP into a managed resource.

```
 File   Edit   Edit_Settings   Menu   Utilities   Compilers   Test   Help
───────────────────────────────────────────────────────────────────────────
EDIT        SDSQ.SFM10.MISC.CNTL(FTPPUT) - 01.47          Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** *************************** Top of Data ****************************
000001 //FTPJOB JOB (BDJM1,1),'FTP to Lab Srvr',MSGCLASS=9,CLASS=A,REGION=4M
000002 //*           NOTIFY=BDJM1,
000003 //*--------------------------------------------------------------*
000004 //* Send file to FTP server                                      *
000005 //*--------------------------------------------------------------*
000006 //*FTPJOB    EXEC PGM=FTP,PARM='-r tls o31v4.sdsusa.lan 8021 (EXIT'
000007 //FTPJOB     EXEC PGM=FTP,PARM='-r tls 10.31.0.1 8021 (EXIT'
000008 //SYSFTPD    DD DISP=SHR,DSN=BCOM2.FTP.DATA.TLS
000009 //*YSFTPD    DD DISP=SHR,DSN=BDJM1.FTP.DATA.TLS
000010 //*YSFTPD    DD DISP=SHR,DSN=BDJM1.FTP.DATA.TLS
000011 //*TPJOB     EXEC PGM=FTP,PARM='-v (EXIT'
000012 //OUTPUT     DD SYSOUT=*
000013 //SYSIN      DD *
000014 user tim
000015 pass ********
000016 bin
000017 cd /u/tin
000018 ls
000019 get /u/bdjm1/dumpfile.bin dump
```

A z/OS FTP batch job that will *fail* because of a simple, one-character typo—but will *not* result in an NMI record being generated by the z/OS FTP client. Thus, there will be no record that this FTP transfer did not take place! Refer to the narrative on the next page.

A QUICK BUT TELLING EXAMPLE OF CRIPPLING BLIND SPOTS IN Z/OS FTP

Consider the relatively simple, but very typical, z/OS FTP batch job example shown on the previous page. In this example, a user named Tim is trying to upload a file to a directory on the remote machine designated "Tim." Tim, however, in the change directory (CD) command on line 17 of the JCL, mistypes the directory name. Instead of "Tim," he types in "Tin". There is no such directory on the remote machine.

Consequently, when the z/OS FTP client sends this bad CD command to the remote FTP server, it will receive an error code. The z/OS FTP client will then abort the job. z/OS will not notify Tim, in real time, that his FTP job aborted. Moreover, there will be no NMI record generated to denote that the transfer did not occur and that the FTP job was abnormally terminated! The z/OS FTP client only generates an NMI record when an actual transfer, i.e. a put or get, is attempted. In this case, the error occurred prior to the put command.

One can now easily visualize and even relate to the chain of events that are likely to unfold. Assume that Tim was uploading this file so that a colleague, at a different location, can use it to complete a high-priority project. At some point Tim will get a call from his colleague to say: "Hey, I am still waiting. What's up?" Tim will claim that he did indeed send the file and that it should have been uploaded hours ago. While still on the phone, Tim scrambles to log onto the remote machine to locate the file. To his chagrin, he is unable to locate it.

Tim then calls the local help desk. They query their logs but do not see anything pertaining to Tim's transfer. There is no confidence-of-success indication because the transfer never took place! They now have to start combing through the job log to find what happened to that job. This takes time. At this stage nobody is sure whether the job was executed—believing that this is the root cause of the problem. Tim, unconvinced, decides that he better log on to TSO and see if he can find the job. He discovers that his job did indeed run. Now he has to look through the SYSOUT to see what transpired. It is then that he finds that a slip on the keyboard was the culprit and that he needs to correct the CD statement and resubmit the job.

Suffice to say Tim is not pleased that it took this much effort to determine why his FTP transfer failed. Neither is his colleague or, for that matter, the help desk personnel. Time was wasted. Productivity was squandered. Deadlines were sacrificed. Tempers frayed. User satisfaction suffered. All because z/OS FTP is in essence an unmanaged service—totally defying what is customarily expected from mission-critical, high-volume system utilities.

Things would have been very different if VFTP had been present.

Authorized users can gain access to the comprehensive and incisive management data collected and collated by VFTP via a very visual, point-and-click, web-browser-based interface. Thus, it would be possible to permit users like Tim, as well as help desk personnel, to have access to VFTP.

With access to VFTP, all that Tim would have had to do when he received the dreaded "Where is the file you were sending me?" was to log onto VFTP and quickly click on the Problem Sessions query. He would have immediately seen his FTP job on the list.

He could then pull up the detailed VFTP Session Activity Log (see page 9) with another click. This log would tell him, instantly, what went wrong. Tim could have done all of this while he was still on the phone with his colleague. He could then have explained what happened. VFTP would have prevented the frustration and the time that was squandered.

39	11Jul2007	14:17:22	EZA1460I Command:
40	11Jul2007	14:17:22	EZA1736I cd /u/tin/file
41	11Jul2007	14:17:22	EZA1701I >>> CWD /u/tin/file
42	11Jul2007	14:17:22	EZA1735I Std Return Code = 07550, Error Code = 00002
43	11Jul2007	14:17:22	550 CWD cmd failed : EDC5129I No such file or directory. (errno2=0x0594003D)
44	11Jul2007	14:17:22	EZA1460I Command:

Expanded view of the entries from the VFTP Session Activity Log with the error clearly highlighted to facilitate instant detection.

Z/OS FTP IS UNMANAGED UNTIL AUGMENTED WITH AN FTP MANAGER

Standard z/OS FTP is, in effect, an unmanaged service because the IBM supplied 'management' features are not comprehensive and cohesive, as shown by the above example. The IBM standard features are:

» Incomplete – SMF/NMI records not generated for all FTP commands.

» Intractable – no automated or conditional retries of failed commands.

» Inexact – inability to selectively apply security criteria to individual FTP commands.

» Incapable – additional third-party software required to exploit server exits and client API.

» Incommunicative – limited options for real-time notifications via email or WTO operator console messages.

» Inefficacious – NMI/SMF records do not provide necessary context and are difficult to categorize for audit purposes.

Given these demonstrable shortcomings, it is easy to see why standard z/OS FTP has to be augmented with a suitable FTP Manager à la VFTP. Without such an FTP manager you will continue to be confronted daily with the following types of problems:

1. Inability to implement meaningful automation, in particular for batch jobs, to overcome transient network outages, override certain return codes, and execute recovery measures or activate contingency options—using conditional IF-THEN-ELSE FTP execution sequences.

2. Difficulty in providing the relevant FTP history records, suitably grouped, to meet current audit and compliance requirements, e.g. HIPAA and Sarbanes-Oxley.

3. Exposure to major security breaches given that specific security criteria cannot be selectively applied to individual FTP commands or file types, on a per-authorized-user basis, in concert with the z/OS SAF security facility (e.g. RACF). Thus, there will be constant dangers such as users with read-only access being able to initiate off-site transfers or users trying to exploit certain functions of the potent z/OS server SITE command.

4. Users, system/network operators, and help desk personnel experiencing productivity and morale sapping delays due to the absence of incisive, real-time and historical FTP monitoring that would let them determine and rectify FTP-related operational issues—quickly and easily. True FTP monitoring would put an end to that plaintive, perennial cry for help: "Can somebody please tell me what happened to that file I was trying to send with FTP?"

With this insight into the management deficiencies of standard z/OS FTP, it is now easy to compile a profile of what a proficient FTP manager needs to be in order to make z/OS FTP into a well-managed, mainframe-class service. The table on page 10 sets out to do just this by categorizing the desired characteristics of a full-function FTP manager in terms of must-have capabilities and value-enhancing features. The FTP manager that you opt for should indubitably possess, without compromise, all the must-have capabilities and quite a few of the value-enhancing features.

VFTP's information-packed and easy-to-follow FTP Session Activity Log.

CHARACTERISTICS OF A PROFICIENT Z/OS FTP MANAGER

IMPERATIVE	HIGHLY DESIRABLE
⊖ Provide automation, auditing, security, and monitoring for *both* the z/OS FTP server and client from within a single unified, consistent framework.	⊖ Browser-based, very visual, point-and-click monitoring with a constantly visible navigation tree and instant drill-down options—that is intuitive and simple to master.
⊖ Extract, synthesize, and collate management data from NMI records, FTP client API, and FTP server exits to ensure total visibility with no potential for blind spots (even when no SMF/NMI records are generated).	⊖ Zero dependence on inefficient and *often inconclusive* (as when transactions are encrypted) data collection techniques such as packet tracing.
⊖ Powerful, but easy to master, FTP control language to realize batch-mode FTP client automation.	⊖ Modifiable queries that can be quickly and precisely targeted to monitor specific types of file, user, or session activity—with the option of saving the queries for later use.
⊖ Ability to selectively apply security criteria to individual FTP commands or file types, on a per-authorized-user basis (with optional date/time criteria), in concert with the z/OS SAF security facility (e.g. RACF).	⊖ Ability to easily locate various z/OS FTP problems with a *single click*—including client and server transfer failures, premature client termination, log-in failures, transfers that ended with low confidence levels, and commands that were rejected by security rules.
⊖ Detailed logging of FTP sessions and transfers for *both real-time* monitoring as well as data for regulation- compliant, off-line auditing.	⊖ Tight integration with RACF, ACF2, and Top Secret.
⊖ Options for notifying users and operators of FTP status and progress via email or operator console messages.	⊖ Customizable summary-level reporting to realize a bird's-eye, network-wide view of FTP usage in terms of who, what, when, and where.
⊖ Data collection that is unimpeded by SSL/TLS encrypted transfers.	⊖ Clean, scalable, low-overhead architecture that relies exclusively on standard IBM provided APIs, exits, and data, and is fully conformant with industry and z/OS standards.
⊖ FTP client API usage that in no way interferes with or compromises the functioning of the client.	⊖ Not dependent on external software such as DB2 and WebSphere.
⊖ Augmentation of IBM's new confidence-of-success indicator to embrace error scenarios not covered by standard z/OS FTP.	⊖ Mainframe management software that is quick and easy to install and maintain.

SDS, a company that has been successfully delivering mainframe software staples since 1982, has made an indelible mark in the mainframe IP management arena with its state-of-the-art, feature-rich, and nimble VitalSigns for IP (VIP). VIP has been in production use since 2003 as a pervasive IP status, problem, and performance monitor. It provides at-a-glance visibility of all the popular IP applications, including FTP.

VIP is not, however, an FTP manager. VitalSigns for FTP is.

VFTP benefits from the experience and expertise that SDS gained with VIP providing mainframe customers with in-depth, real-life, real-time IP management. Consequently, the product objectives for VFTP were built around explicit customer wishes, requests, and expectations as to z/OS FTP monitoring, automation, and control. VFTP sets out to satisfy all the characteristics of a proficient z/OS FTP manager as enumerated in the table on the previous page.

VFTP modernizes z/OS FTP and transforms it into a mainframe-class, secure, mission-critical service. With VFTP, z/OS FTP can finally fulfill the competitive, compliance, security, and user satisfaction demands now confronting enterprises around the world.

VFTP relies on an agent/server architecture with the agent, server, and VFTP database all being z/OS-based in VFTP.

The VFTP agent assimilates data, in real time, from both the z/OS FTP client and server via NMI records, server exits, and the FTP client API (using VFTP's FTP-client wrapper).

All the data gathered by the VFTP agent is maintained on the VFTP database.

The VFTP server collates, analyzes, structures, and formats this data for on-the-fly consumption by operators and users—who access and query the VFTP server through a web-browser-based GUI.

VFTP's *architecture* is discussed further on page 19. VFTP is designed to work with IBM's z/OS FTP client and server.

A well-known U.S. Secretary of Defense once famously said: "There are known knowns. There are known unknowns. But there are also unknown unknowns." Heeding this caution, VFTP ensures that when it comes to z/OS FTP there will no longer be any unknowns, whether known or unknown!

---

1. Easy-to-master and versatile VFTP FTP Control Language (FCL) to automate z/OS FTP client batch mode processing. FCL eliminates the hitherto need for manual intervention whenever there is a glitch in FTP client operations—even if the problem was due to a transient network outage. With FCL it is now possible to implement conditional FTP client execution sequences based on the familiar IF-THEN-ELSE syntax.

    FCL permits the execution of FTP commands to be contingent on the outcome of the previous command, responses received from the server, or return codes generated by the FTP client. FCL makes it possible to:

    » Retry failed transfers on a controlled basis.

    » Determine which failures, under what conditions, warrant recovery, and what the recovery steps should be—thus precluding the squandering of resources on futile or inconsequential retry efforts.

    » Log germane state-of-play bulletins or detailed error messages to the system operator console using WTO commands.

    » Send email to designated personnel to notify them of any FTP aberrations that might jeopardize a file transfer.

    » Maintain an audit trail, at the system console, of all FTP transfers performed or those that failed to complete.

    FCL statements, denoted by a ";!" prefix, can be freely interspersed with FTP commands—as shown in the FTP/FCL sequence on the following page. The use of FCL does not in any way hinder the execution or modify the behavior of FTP commands. The processing of FCL statements to control the flow of FTP commands is performed by the VFTP client wrapper, which interacts with the z/OS FTP client via the IBM supplied API.

    FCL ensures that z/OS FTP batch jobs can now enjoy the level of automation expected from a mainframe utility.

2. Tight integration with RACF, ACF2, and Top Secret, so that z/OS FTP can now be treated as a genuine secure resource. With VFTP it is now possible to selectively apply security criteria to individual FTP commands or file types, on a per-authorized-user basis.

    With VFTP it is possible to give a user read access to a dataset but preclude that dataset from being transferred to a remote host.

    Similarly, it would be possible to give a group of users the right to transfer sequential files but not JES files. Individual FTP commands or specific features of an FTP command can also be selectively disallowed to provide different user groups with customized, controlled FTP capabilities.

```
//VFTP02DJM JOB (BDJM1,1),'VFTP FCL',MSGCLASS=X,CLASS=A,REGION=4M
//*-------------------------------------------------------------*
//* VFTPTEST: RUN VFTP FCL TESTS                                *
//*-------------------------------------------------------------*
//*
//SET1 SET NETRC=BJPC1.VFTPFCL.CNTL(NETRC)
//*
//JS0010 EXEC PGM=FTP
//NETRC DD DISP=SHR,DSN=&NETRC
//SFCOUT DD SYSOUT=* VFTP INPUT LISTING
//OUTPUT DD SYSOUT=* FTP MESSAGE LOG
//SFCEMAIL DD SYSOUT=(C,SMTP) EMAIL
//SYSIN DD *
lcd 'bjpc1.sfmfcl.cntl'
;! if local testin not exist
;!     do
;!         display e last_server_cc 'testin member not found'
;!         mail dmruz@sdsusa.com e last_server_cc +
;!             'bjpc1.sfmfcl.cntl(testin) member not found' log
;!         set step_cc = 8
;!         cancel
;!     end
cd /u/bjpc1
;! if remote testout exist
;!     do
;!         delete testout
;!     end
put 'bjpc1.sfmfcl.cntl(testin)' testout
;! if last_client_cc > 0
;!     do
;!         display e last_client_cc 'get for testin failed'
;!         mail dmruz@sdsusa.com e 'get for testin failed' log
;!         set step_cc = 64
;!     end
;! else
;!     do
;!         display i last_server_cc 'testout updated from testin'
;!         mail dmruz@sdsusa.com i last_server_cc +
;!             'testout updated from testin' log
;!         set step_cc = 0 ; reset to successful cc
;!     end
close
quit
//
```

VFTP's easy-to-master and versatile VFTP FTP Control Language (FCL).

3. Extend and supplement the z/OS FTP confidence-of-success indicator so that it encompasses all FTP attempts—thereby eradicating major black holes in the standard IBM offering which does not adequately handle many FTP error scenarios. The inclusive, VFTP-augmented, confidence-of-success indicators, displayed using impossible-to-miss red/amber/green tokens on the VFTP Session Activity Log, can now be used as a quick and easy way to monitor the progress of FTP transfers and determine follow-up actions.

4. Incisive and resourceful use of the z/OS FTP client API (via the VFTP client wrapper) and specific FTP server exits (e.g. FTCHKCMD and FTPOSTPR[1]) to deliver unprecedented visibility into and control of z/OS FTP operations. It is through the use of the client API and server exits that VFTP is able to provide:

» Details of all FTP commands attempted, successfully or otherwise—independent of whether an NMI/SMF record was generated for that command.

```
31May2007 16:50:56  ■ Confidence=High for STOR of BSFM2.ZAP364
31May2007 16:51:00    --> PORT 192,168,10,16,19,50
31May2007 16:51:00    --> STOR ZAP480
31May2007 16:51:01    <-- 250 Transfer completed successfully.
31May2007 16:51:01    2460 bytes received from 192.168.10.16
31May2007 16:51:01  ■ Confidence=High for STOR of BSFM2.ZAP480
31May2007 16:51:11    --> CWD 'sys1.'
31May2007 16:51:11  ■ SFM security rule rejects CWD command
31May2007 16:52:12    --> SITE filetype=jes
31May2007 16:52:12  ■ SFM security rule rejects FILETYPE parameter
31May2007 16:52:42    --> PORT 192,168,10,16,19,51
31May2007 16:52:42    --> RETR /u/bssjl/sfm/SFM_ENV
31May2007 16:52:42    <-- 250 Transfer completed successfully.
31May2007 16:52:42    484 bytes sent to 192.168.10.16
31May2007 16:52:42    Confidence=Unknown for RETR of /u/bssjl/sfm/SFM_ENV
31May2007 16:52:52    --> PORT 192,168,10,16,19,52
31May2007 16:52:52    --> RETR /u/bssjl/sfm/SFM_OUT
31May2007 16:52:52  ■ <-- 550 Command RETR fails: /u/bssjl/sfm/SFM_OUT does not exist.
31May2007 16:52:52    30 bytes sent to 192.168.10.16
31May2007 16:52:52  ■ Confidence=Low for RETR of /u/bssjl/sfm/SFM_OUT
```

The inclusive, VFTP augmented confidence-of-success indicators,
without the black holes found in the standard offering.

---

[1] FTCHKCMD permits the acceptance or rejection of individual FTP commands while FTPOSTPR is the FTP post-processing exit.

» FCL to automate FTP batch jobs—as discussed in #1 above.

» SAF-based security criteria for individual FTP commands, command features, and file types—#2 above.

» Augmented confidence-of-success indicators—#3 above.

During the VFTP development cycle, SDS discovered a few undocumented anomalies in the z/OS FTP client's behavior in some specific scenarios when the client API was in use. To ensure that VFTP does not in any way alter the FTP client's behavior, VFTP temporarily suspends using the client API when it detects the occurrence of a problem scenario.

5. Total compatibility with z/OS FTP SSL/TLS mode operation. Unlike some of its competitors, all of VFTP's facilities, whether related to monitoring, auditing, security or automation, support encrypted file transfers. Moreover, whether dealing with encrypted or unencrypted transfers, VFTP works the same and provides the same level of management information. VFTP logs complete information on the SSL/TLS state of each session and transfer. Its navigation bar will show you, with a single click, which transfers or sessions were secured or unsecured, making it easy for you to monitor compliance to company standards.

VFTP is well suited for use in either all-encrypted, secure FTP environments or mixed-mode environments that permit both encrypted and unencrypted transfers. Encrypted transfers will become essential as enterprises realize the very real dangers of unencrypted FTP. VFTP, given that its mode of operation stays the same, will facilitate smooth, orderly migration to SSL/TLS mode FTP transfers.

6. Exploiting the unimpeded z/OS FTP visibility possible via the client API and the server exits, VFTP provides a unique, all-encompassing, end-to-end audit trail that contains details of all attempted FTP sessions and transfers. With a VFTP audit trail there will be no omissions just because an FTP session did not result in an actual transfer—as would have been the case with the mistyped CD command discussed on page 6.

A VFTP audit trail will always include the following details for both the z/OS FTP client and server:

» Who transferred what to/from where, when, and how.

» Completion status of the transfer.

» Authorization credentials for the transfer, i.e. was it an authorized transfer?

VFTP provides a holistic view that presents FTP activity in the context of the FTP sessions that initiated it, so that transfers are not seen as disconnected events. VFTP correlates all of the wealth of FTP data it monitors back to the FTP session. Hence, whether you are investigating a transfer, a log-in failure, a dataset that was deleted or renamed with FTP, or an entry in the VFTP message log, an exhaustive log of everything that happened in the course of the session is never more than one or two mouse clicks away.

With VFTP, IT auditors or system administrators can easily delve into the complete, bi-directional history of a z/OS system—and even have the option to do so with all the audit trail records grouped together by user ID, FTP session number, LPAR, or sysplex. Suffice to say that VFTP's audit trail easily satisfies the requirements of today's stringent audit and IT compliance regulations, e.g. HIPAA and Sarbanes-Oxley.

7. Access to VFTP's information-packed database, via quickly customizable queries, with graphical output, to be able to gain instant, at-a-glance visibility into any and all aspects of z/OS FTP related activity—on a user, session, or file-name basis. The screen shot on the next page shows how queries can be created. Specific summary information can also be requested.

VFTP also comes pre-configured with a set of standard queries which can be used to obtain:
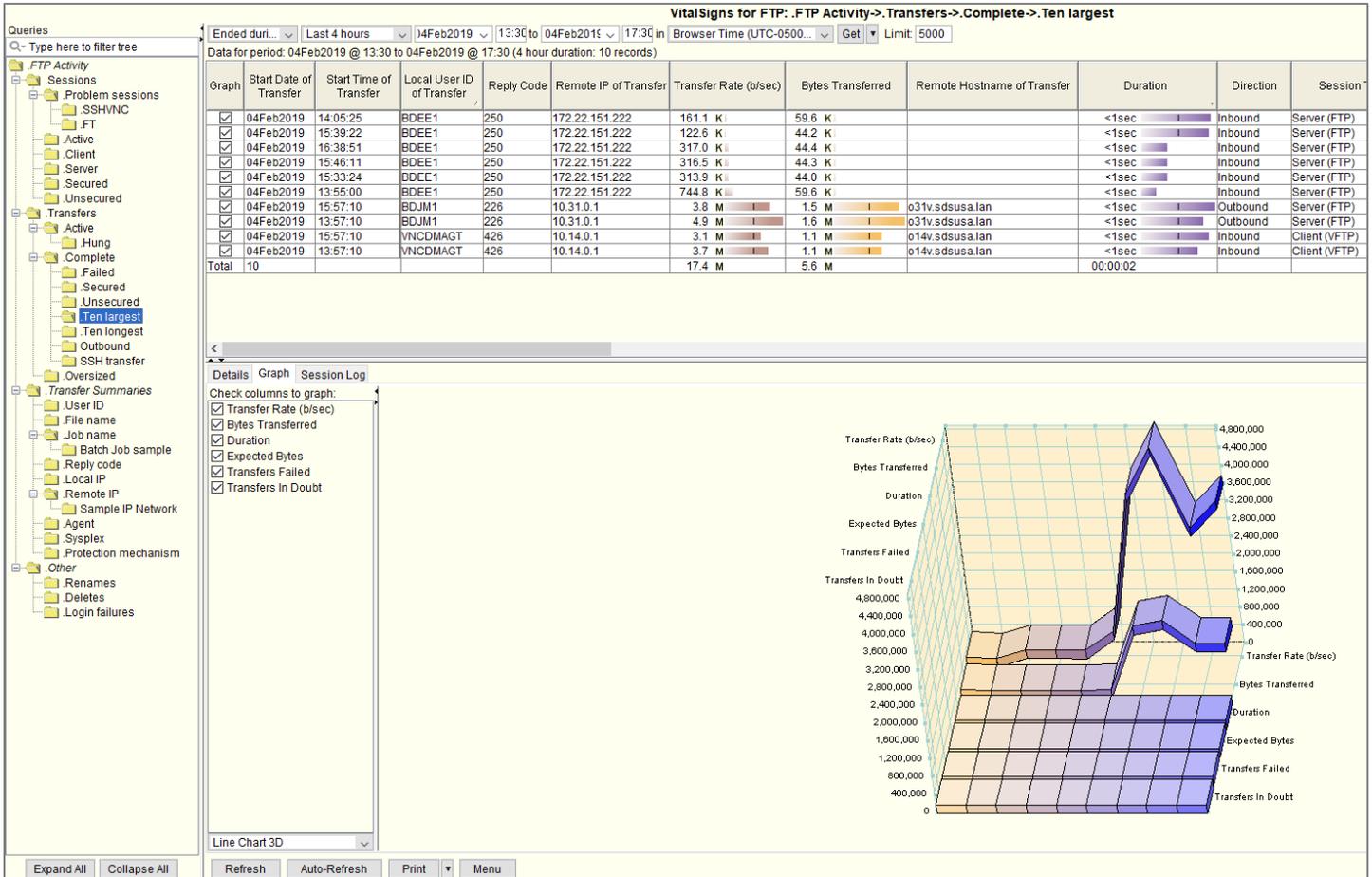
• 10 most-active FTP users	• FTP jobs submitted
• Failed transfers	• Suspect transfers
• Size of files transferred	• Elapsed time for transfers
• Server log ons that failed	

VFTP provides help desk personnel, system operators, operations analysts, and network administrators with all the pertinent data they require to troubleshoot FTP issues, research FTP activity, and proactively monitor FTP operations.

8. Powerful automated operations enablement capabilities which enable automated operations environments to be controlled and driven via meaningful, easily-parsed system console messages whenever a critical event occurs. VFTP can be easily configured to write messages to the system console for any of the following events:

• Client transfer completion (all or just the failures)

- Server transfer completion (all or just the failures)

- Client step termination (all or just the failures)

- Server log-in failures

- Server commands rejected due to security rules

- Messages from the FTP client FCL

VFTP's incisive and versatile FTP Activity Log—in this instance displaying details of the 10 largest, completed transfers. Further details on specific transfers can be readily obtained just by clicking on the entry for a transfer.

9. Real-time, watch dog-mode surveillance capability of all FTP log ons, if desired, to enhance both security and productivity by permitting system/network operators to:

- Detect when authorized users are experiencing trouble and proactively intervene, via instant message (IM), email, or phone (outside of VFTP), to help them successfully log on—avoiding wasted time, frayed tempers, and project delays.

- Observe intrusion attempts by unauthorized users and quickly take action to preclude them from gaining access to z/OS assets.

10. Problem-sessions query capability with a unique ability to neatly pull together many different types of FTP problems to ensure that any kind of FTP problem can be located and identified—quickly and effortlessly. VFTP provides a capacity to uncover transfer failures, client steps that terminated in error, server log-on failures, transfers that ended with low confidence levels, commands that were rejected by security rules, and more—all with but a single mouse click.
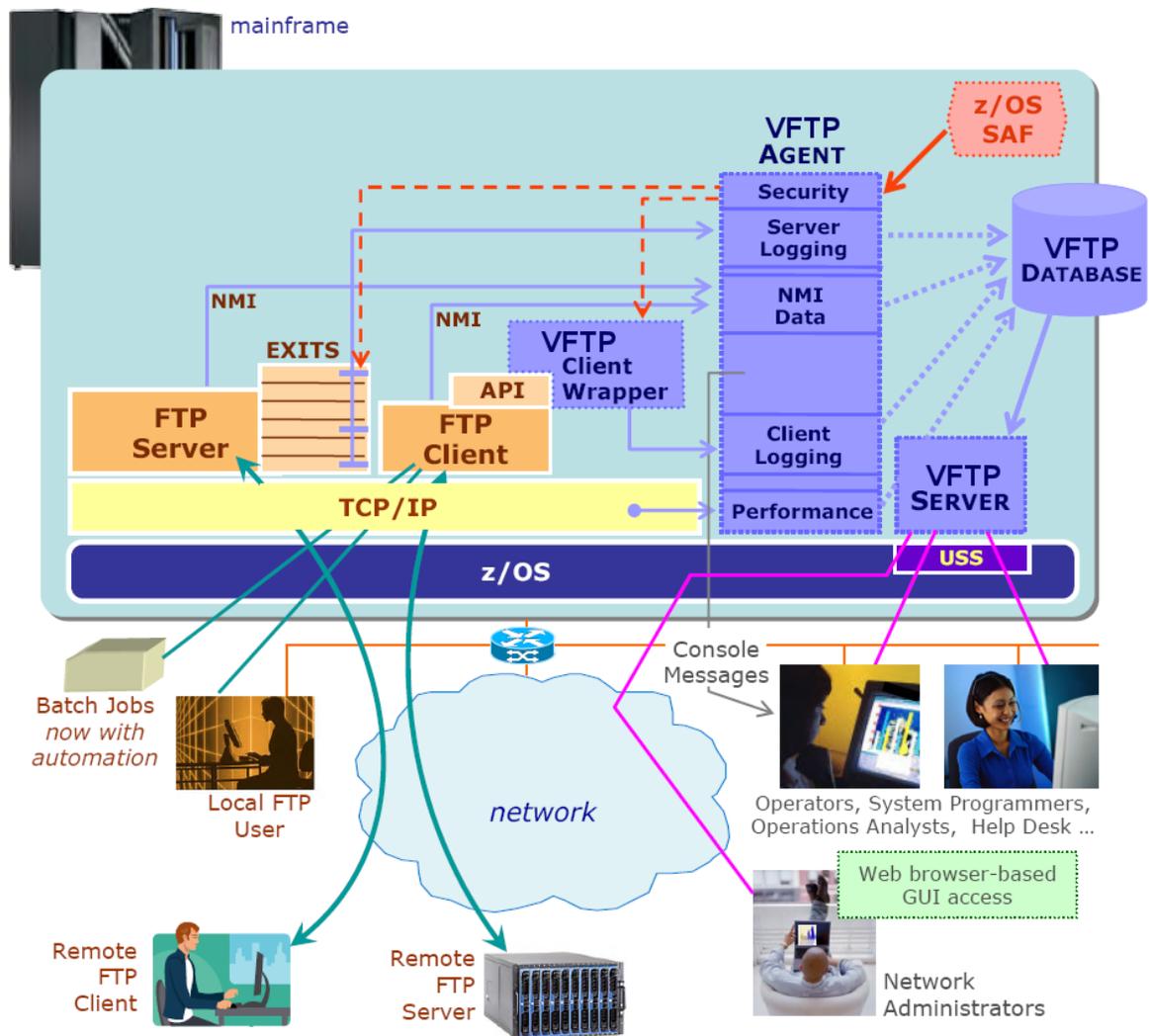
This list of 10 standout features of VFTP is but an attempt to whet the appetite. It does not, by any means, cover all the advantages that an enterprise can derive from using VFTP. Given its powerful customization options and its multi-faceted capabilities covering automation, security, monitoring, and auditing, VFTP can be readily tailored to meet the specific requirements of a particular z/OS environment. It is not meant to be a rigid, one-size-fits-all product. Instead, customers can discover unexpected synergies by mixing and matching VFTP's various features to best suit their ongoing demands—e.g. creating customized queries to check on the success of automated batch transfers.

SDS, very confident of VFTP's strengths, lets companies try out VFTP for free against the customer's actual z/OS FTP traffic. SDS will also provide all the necessary documentation and technical support. This is a genuine, win-win proposition. So here is the suggested game plan. Look through the VFTP architecture section that follows, contact SDS, schedule a web demonstration if you wish—or even skip that step and just go with the free trial.

VFTP consists of four distinct components:

1. The VFTP agent, which runs in its own z/OS address space. The VFTP agent is responsible for assimilating all the z/OS FTP client and server information sought by VFTP. It does so, as shown below, using four separate mechanisms: NMI record access, FTP server exit logging, client activity monitoring via the client API, and extraction of FTP performance data from the TCP/IP stack. The VFTP agent is also responsible for enforcing the selective per FTP command security criteria using SAF rules. The VFTP agent implements this security on the server via the FTP command exit.

VFTP's z/OS-oriented agent-server architecture.

2. The VFTP server, which runs on z/OS USS. The VFTP server is the focal point through which VFTP users gain access, via a standard web browser, to all the FTP activity-related data and reports maintained by VFTP.

3. The VFTP client wrapper, which interacts with the z/OS FTP client via the IBM supplied API. It enables VFTP to precisely monitor every action performed by the client including the processing of all FTP commands and log-on attempts. It is the client wrapper that enables VFTP to provide a plethora of information that goes well beyond what is available just by analyzing the NMI records generated by the client. The FCL-based client automation and per- FTP-command security enforcement is also realized via the client wrapper.

4. The VFTP database, like the VFTP server, runs on z/OS USS and is based on open systems technology. It acts as a secure repository for all of the z/OS FTP server and client data collected by the VFTP agent. The data maintained in this database is accessed via the VFTP server.

VFTP works with the IBM FTP server and client. The VFTP server, which runs on USS, requires the IBM z/OS Java SDK; check with SDS for information for your configuration.

VFTP is robust and easy to install. Installation just requires the execution of a single job stream that installs the VFTP agent, VFTP server, and VFTP database.

The z/OS TCP/IP profile needs to specify "NETMONITOR SMFSERVICE," while integration with z/OS is dependent on the presence of security offerings, such as RACF, ACF2 or Top Secret, that support IBM's z/OS SAF interface.

THE BOTTOM LINE

Standard z/OS FTP–without a full-function FTP manager such as VitalSigns for FTP–cannot provide the automation, monitoring, security or auditing capabilities expected of a high-volume, mission-critical mainframe utility. Unmanaged z/OS FTP will compromise mainframe operations—and in this instance it is not a question of 'if' but rather 'when.' VFTP is a thoughtfully architected, z/OS-specific solution that deftly rectifies the deficiencies of standard FTP. With VFTP in place, augmenting z/OS FTP, there will no longer be any unknowns, security exposures, compliance shortfalls or operational setbacks due to unautomated transfers. VFTP transforms standard z/OS FTP into a highly secure, mission-critical mainframe utility that meets auditing and compliance requirements.

API	Application Program Interface, i.e. programmatic access to an application—in this instance the IBM z/OS FTP client.
CD	FTP's Change (remote) Directory command.
Exits	IBM-sanctioned mechanisms whereby specific application-provided routines, accessible to other software, are activated when certain application events occur—for example, in the context of the z/OS FTP server, prior to the execution of each FTP command.
FCL	VFTP's easy-to-master but versatile FTP Control Language for automating FTP batch-mode processing by the z/OS FTP client.
FTP	File Transfer Protocol.
FTP Confidence of Success	IBM's measurement that provides an indication of the success of individual file transfers. Support for this is provided by the CHKCONFIDENCE TRUE parameter in the FTP data file. For certain types of transfers, the z/OS FTP client and server can be configured to perform additional checks and report a level of confidence that transfers have completed successfully. This is designed to provide an additional safeguard against data loss by including checks not provided for in the FTP protocol.
LCD	FTP's Change Local Directory command.
NMI	IBM's Network Management Interface for obtaining certain management related data—in real time.
SAF	Security Authentication Facility, an IBM feature that allows products like VFTP to interface with various z/OS security products, e.g. RACF, ACF2 or Top Secret.
SMF	IBM's long-standing System Management Facilities (SMF) mechanism for collecting mainframe-related management information via application generated SMF records.
SSL	Secure Sockets Layer—a widely used mechanism for providing client/server authentication and data encryption.
TLS	Transport Layer Security—the successor to SSL.
VFTP	VitalSigns for FTP, SDS' FTP manager for transforming z/OS FTP from an unmanaged service into managed service complete with extensive automation, monitoring, security, and auditing capabilities.