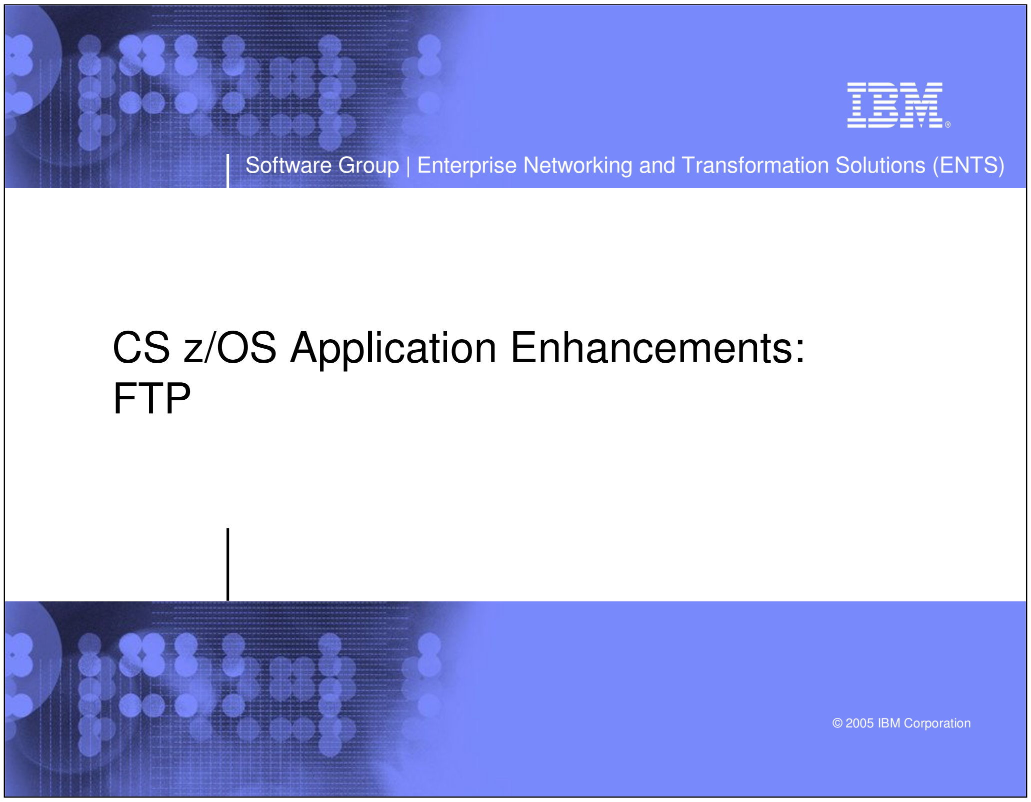
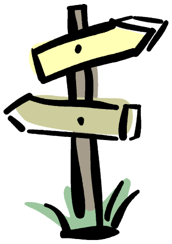


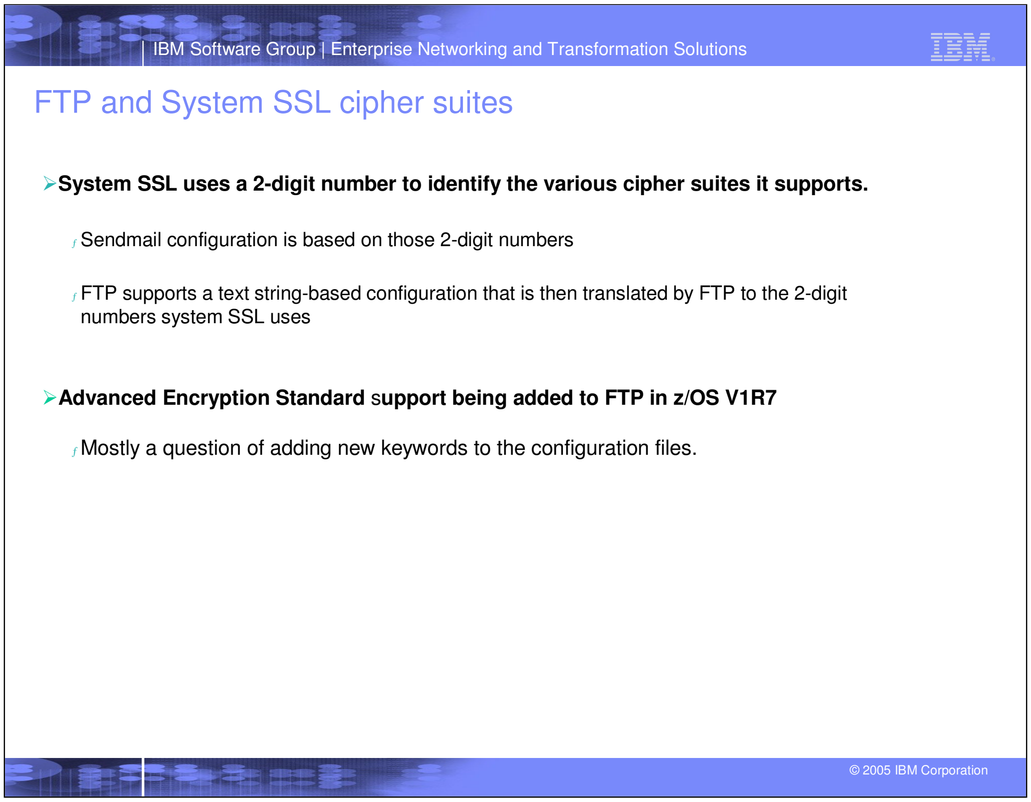
Software Group | Enterprise Networking and Transformation Solutions (ENTS)

CS z/OS Application Enhancements: FTP

Agenda

- AES encryption support
- Delegated RACF resource profiles for TLS FTP
- FTP support for mixed-case RACF passwords
- FTP JES SAPI interface changes
- FTP client API in C programming language
- FTP data transfer reliability feedback
- FTP configurable end-of-line character
- Enable/disable extended directory search





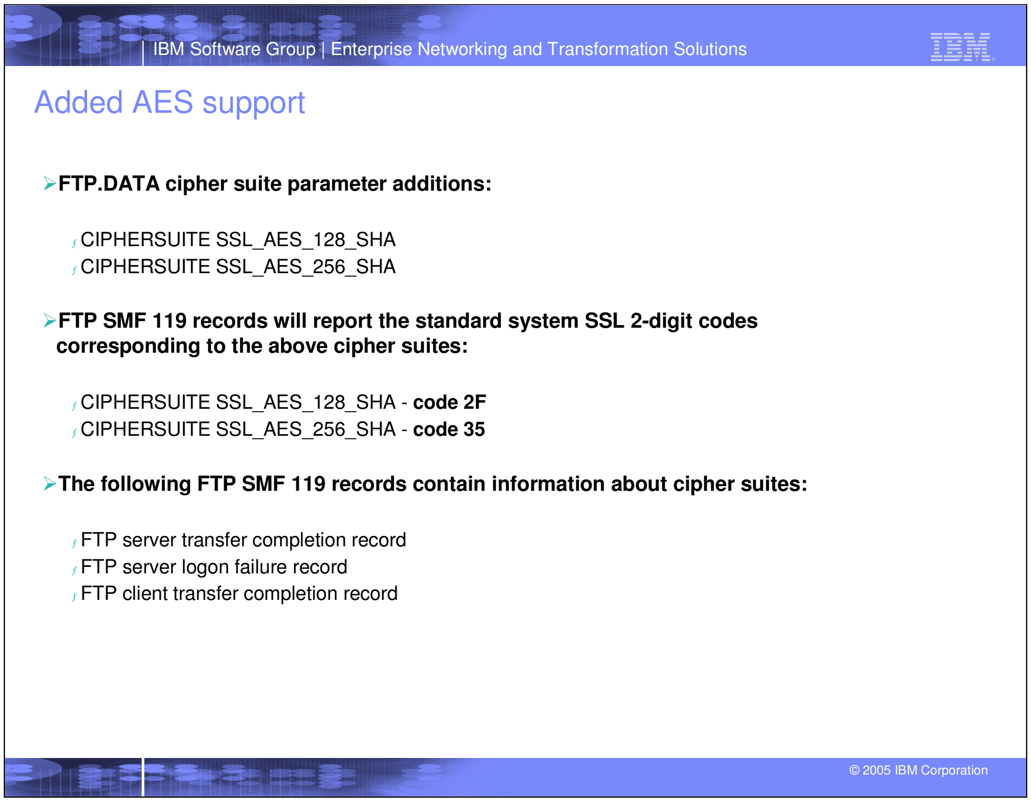
FTP and System SSL cipher suites

- **System SSL uses a 2-digit number to identify the various cipher suites it supports.**
  - Sendmail configuration is based on those 2-digit numbers
  - FTP supports a text string-based configuration that is then translated by FTP to the 2-digit numbers system SSL uses
- **Advanced Encryption Standard support being added to FTP in z/OS V1R7**
  - Mostly a question of adding new keywords to the configuration files.

Supported System SSL cipher suites

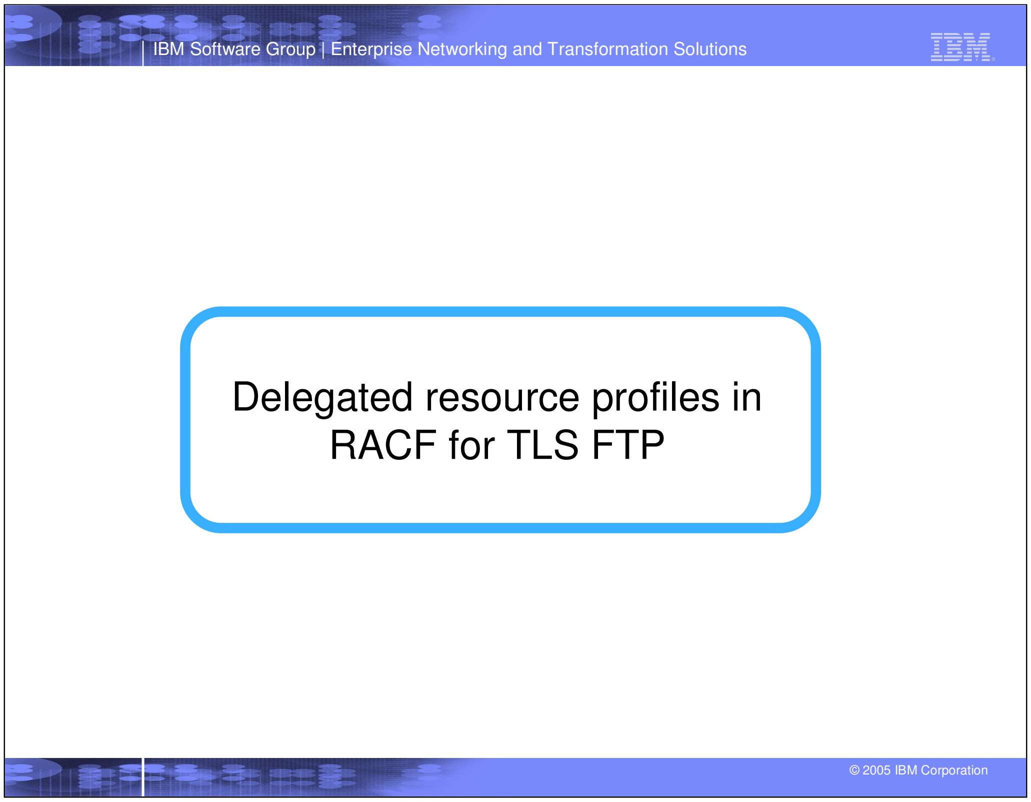
FTP supports the following cipher suites:

- SSL_DES_SHA
- SSL_3DES_SHA
- SSL_NULL_MD5
- SSL_NULL_SHA
- SSL_RC2_MD5_EX
- SSL_RC4_MD5
- SSL_RC4_MD5_EX
- SSL_AES_128_SHA
- SSL_AES_256_SHA



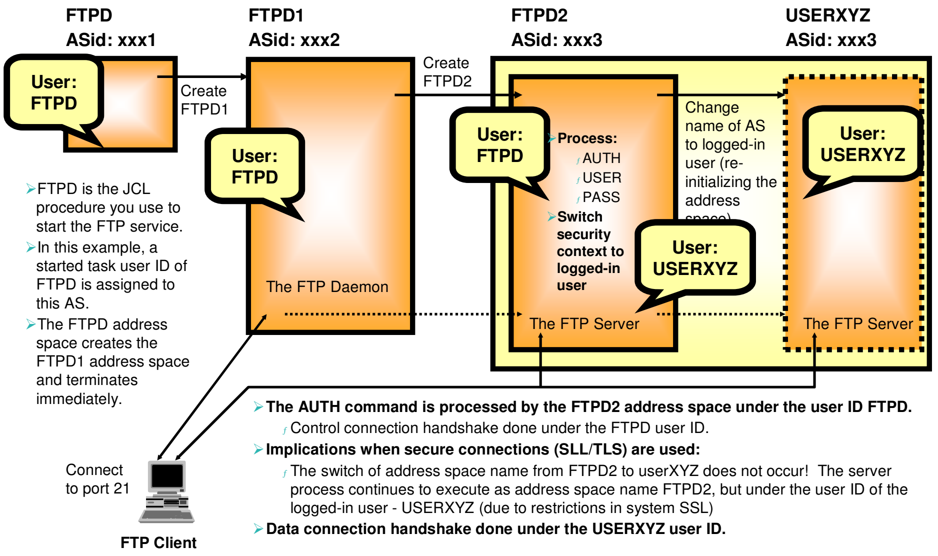
Added AES support

- **FTP.DATA cipher suite parameter additions:**
  - CIPHERSUITE SSL_AES_128_SHA
  - CIPHERSUITE SSL_AES_256_SHA
- **FTP SMF 119 records will report the standard system SSL 2-digit codes corresponding to the above cipher suites:**
  - CIPHERSUITE SSL_AES_128_SHA - **code 2F**
  - CIPHERSUITE SSL_AES_256_SHA - **code 35**
- **The following FTP SMF 119 records contain information about cipher suites:**
  - FTP server transfer completion record
  - FTP server logon failure record
  - FTP client transfer completion record



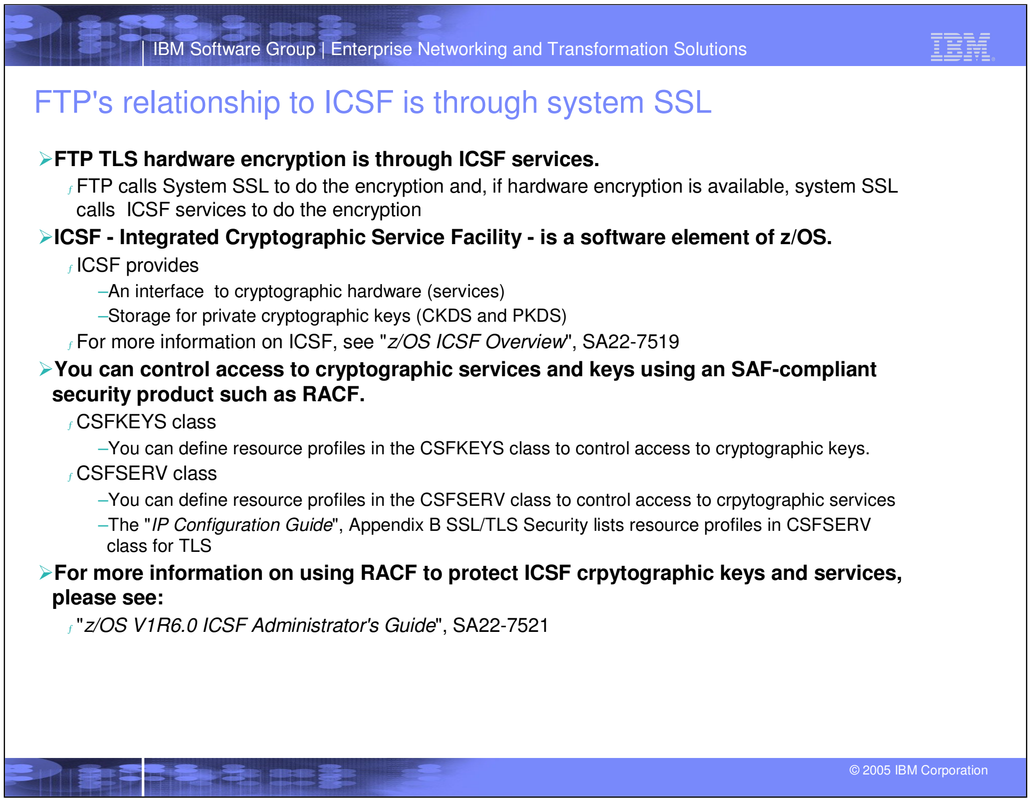
Delegated resource profiles in RACF for TLS FTP

The FTP server and its related address spaces - SSL/TLS



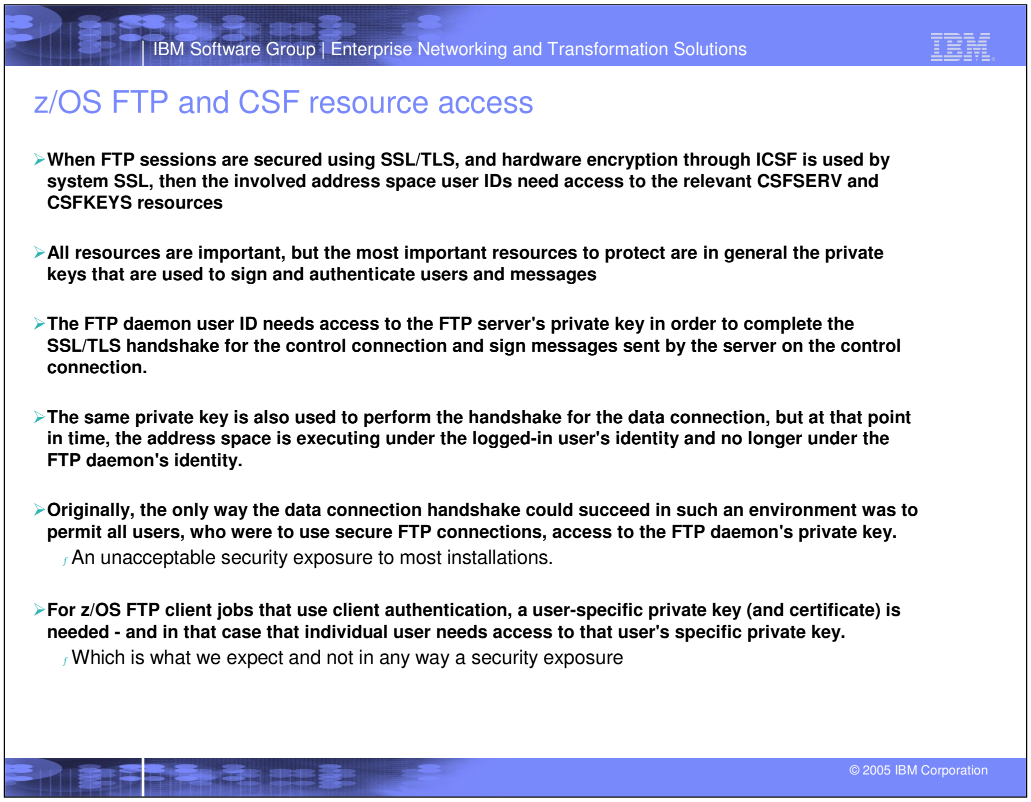
- FTPD is the JCL procedure you use to start the FTP service.
- In this example, a started task user ID of FTPD is assigned to this AS.
- The FTPD address space creates the FTPD1 address space and terminates immediately.

- **The AUTH command is processed by the FTPD2 address space under the user ID FTPD.**
  - Control connection handshake done under the FTPD user ID.
- **Implications when secure connections (SLL/TLS) are used:**
  - The switch of address space name from FTPD2 to userXYZ does not occur! The server process continues to execute as address space name FTPD2, but under the user ID of the logged-in user - USERXYZ (due to restrictions in system SSL)
- **Data connection handshake done under the USERXYZ user ID.**



FTP's relationship to ICSF is through system SSL

- **FTP TLS hardware encryption is through ICSF services.**
  - FTP calls System SSL to do the encryption and, if hardware encryption is available, system SSL calls ICSF services to do the encryption
- **ICSF - Integrated Cryptographic Service Facility - is a software element of z/OS.**
  - ICSF provides
    - An interface to cryptographic hardware (services)
    - Storage for private cryptographic keys (CKDS and PKDS)
  - For more information on ICSF, see "*z/OS ICSF Overview*", SA22-7519
- **You can control access to cryptographic services and keys using an SAF-compliant security product such as RACF.**
  - CSFKEYS class
    - You can define resource profiles in the CSFKEYS class to control access to cryptographic keys.
  - CSFSERV class
    - You can define resource profiles in the CSFSERV class to control access to crpytographic services
    - The "*IP Configuration Guide*", Appendix B SSL/TLS Security lists resource profiles in CSFSERV class for TLS
- **For more information on using RACF to protect ICSF crpytographic keys and services, please see:**
  - "*z/OS V1R6.0 ICSF Administrator's Guide*", SA22-7521



z/OS FTP and CSF resource access

- **When FTP sessions are secured using SSL/TLS, and hardware encryption through ICSF is used by system SSL, then the involved address space user IDs need access to the relevant CSFSERV and CSFKEYS resources**

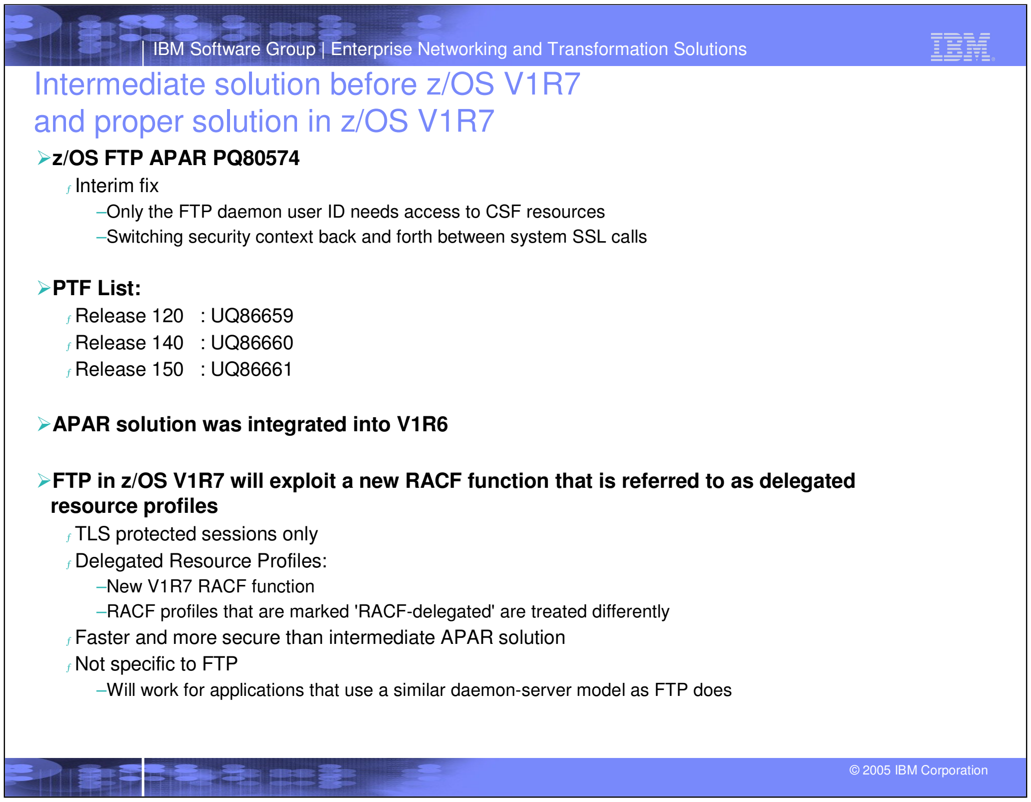
- **All resources are important, but the most important resources to protect are in general the private keys that are used to sign and authenticate users and messages**

- **The FTP daemon user ID needs access to the FTP server's private key in order to complete the SSL/TLS handshake for the control connection and sign messages sent by the server on the control connection.**

- **The same private key is also used to perform the handshake for the data connection, but at that point in time, the address space is executing under the logged-in user's identity and no longer under the FTP daemon's identity.**

- **Originally, the only way the data connection handshake could succeed in such an environment was to permit all users, who were to use secure FTP connections, access to the FTP daemon's private key.**
  - An unacceptable security exposure to most installations.

- **For z/OS FTP client jobs that use client authentication, a user-specific private key (and certificate) is needed - and in that case that individual user needs access to that user's specific private key.**
  - Which is what we expect and not in any way a security exposure



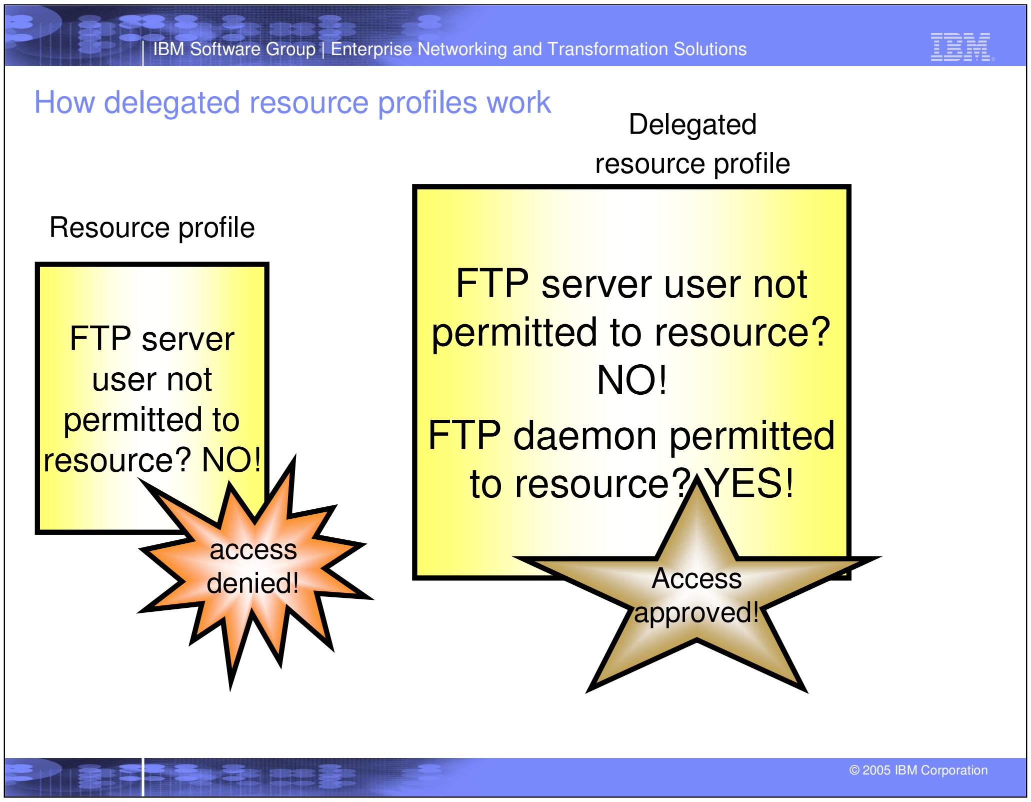
Intermediate solution before z/OS V1R7 and proper solution in z/OS V1R7

- **z/OS FTP APAR PQ80574**
  - Interim fix
    - Only the FTP daemon user ID needs access to CSF resources
    - Switching security context back and forth between system SSL calls

- **PTF List:**
  - Release 120 : UQ86659
  - Release 140 : UQ86660
  - Release 150 : UQ86661

- **APAR solution was integrated into V1R6**

- **FTP in z/OS V1R7 will exploit a new RACF function that is referred to as delegated resource profiles**
  - TLS protected sessions only
  - Delegated Resource Profiles:
    - New V1R7 RACF function
    - RACF profiles that are marked 'RACF-delegated' are treated differently
  - Faster and more secure than intermediate APAR solution
  - Not specific to FTP
    - Will work for applications that use a similar daemon-server model as FTP does



How delegated resource profiles work

Resource profile

FTP server user not permitted to resource? NO!

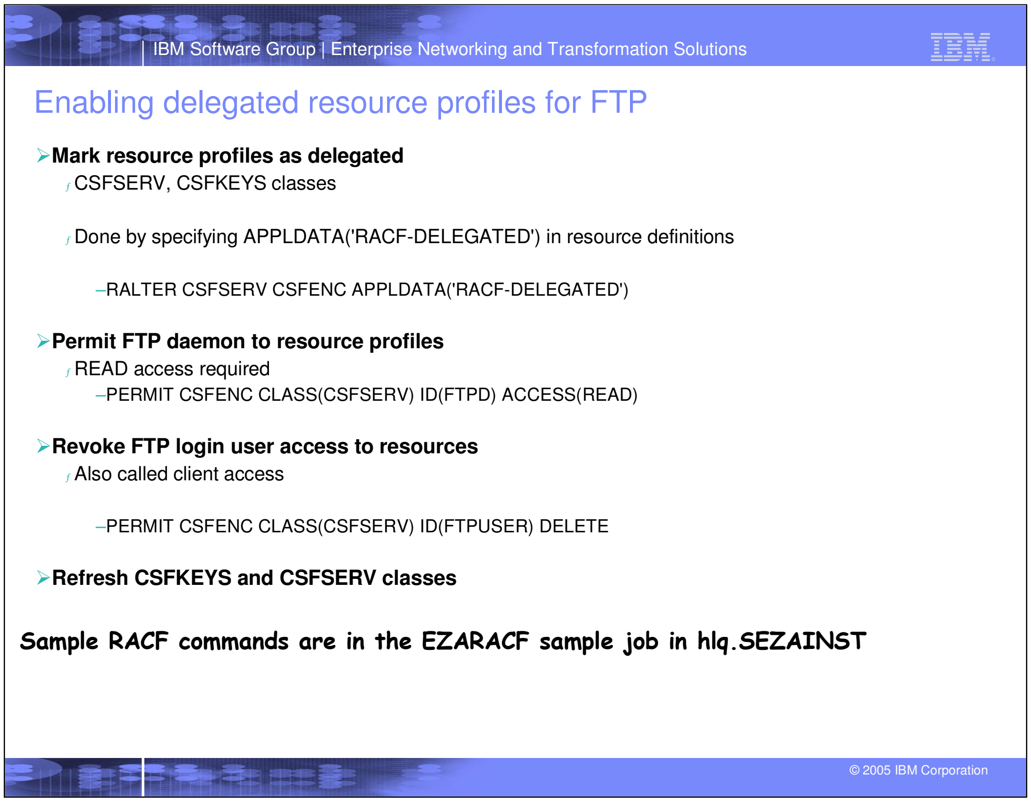
access denied!

Delegated resource profile

FTP server user not permitted to resource? NO!

FTP daemon permitted to resource? YES!

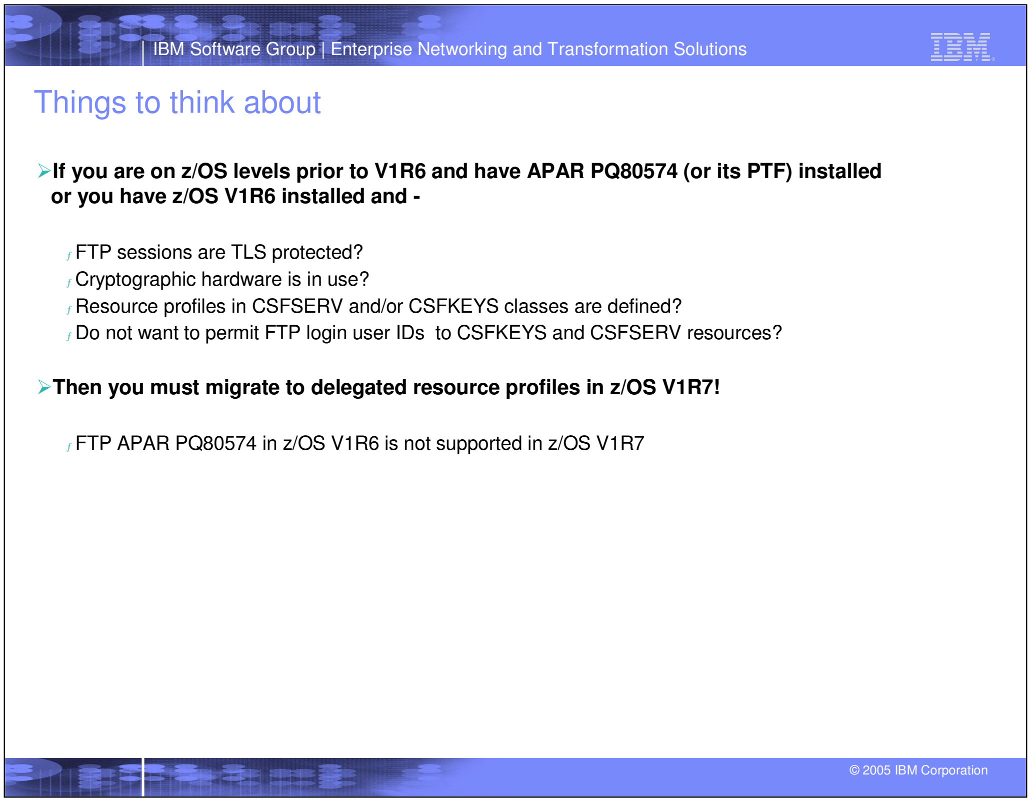
Access approved!



Enabling delegated resource profiles for FTP

- **Mark resource profiles as delegated**
  - CSFSERV, CSFKEYS classes
  - Done by specifying APPLDATA('RACF-DELEGATED') in resource definitions
    - –RALTER CSFSERV CSFENC APPLDATA('RACF-DELEGATED')
- **Permit FTP daemon to resource profiles**
  - READ access required
    - –PERMIT CSFENC CLASS(CSFSERV) ID(FTPD) ACCESS(READ)
- **Revoke FTP login user access to resources**
  - Also called client access
    - –PERMIT CSFENC CLASS(CSFSERV) ID(FTPUSER) DELETE
- **Refresh CSFKEYS and CSFSERV classes**

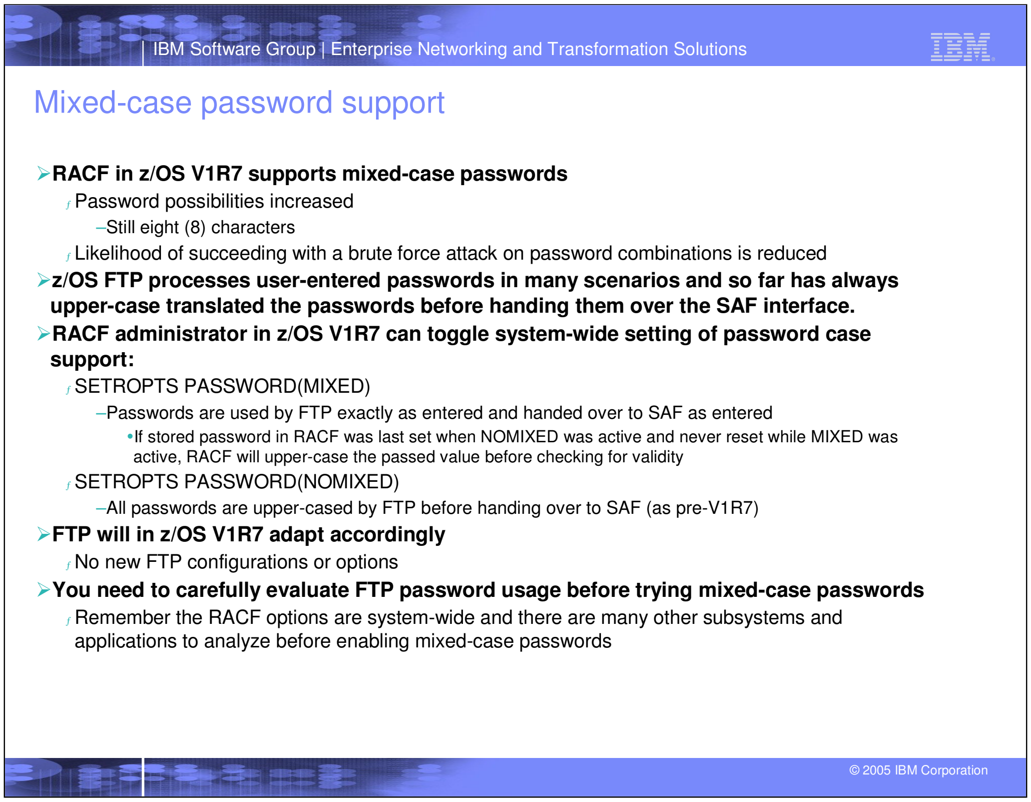
Sample RACF commands are in the EZARACF sample job in hlq.SEZAINST



Things to think about

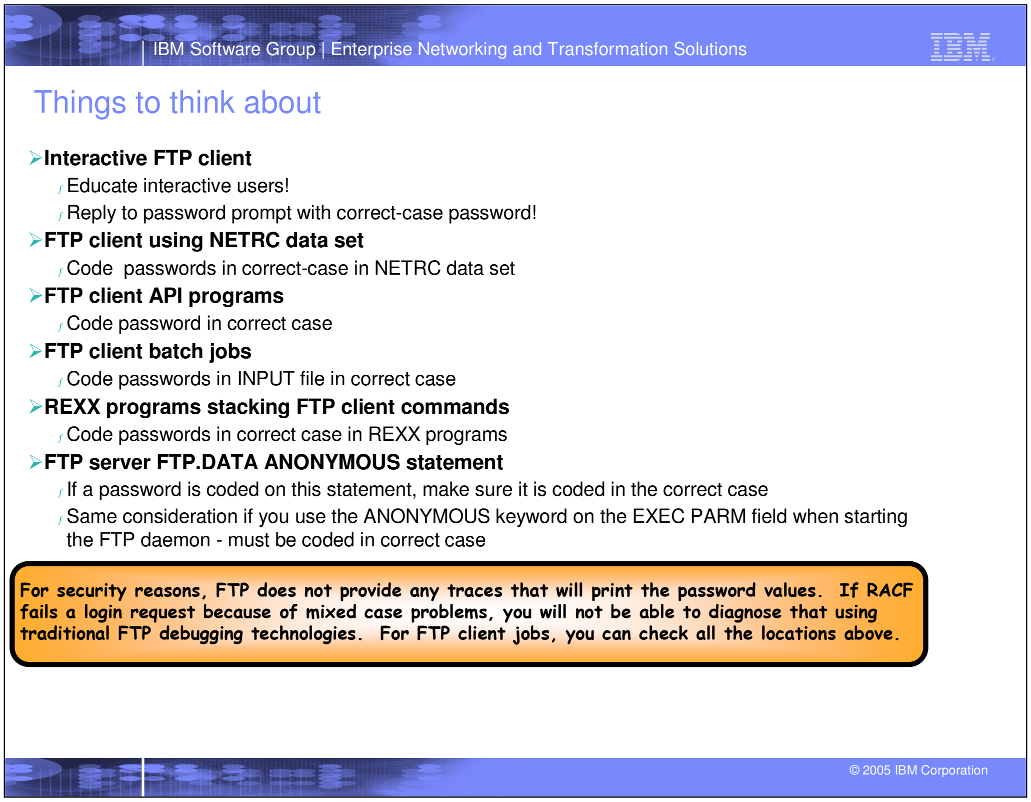
- **If you are on z/OS levels prior to V1R6 and have APAR PQ80574 (or its PTF) installed or you have z/OS V1R6 installed and -**
  - FTP sessions are TLS protected?
  - Cryptographic hardware is in use?
  - Resource profiles in CSFSERV and/or CSFKEYS classes are defined?
  - Do not want to permit FTP login user IDs to CSFKEYS and CSFSERV resources?
- **Then you must migrate to delegated resource profiles in z/OS V1R7!**
  - FTP APAR PQ80574 in z/OS V1R6 is not supported in z/OS V1R7

FTP and mixed-case RACF passwords



Mixed-case password support

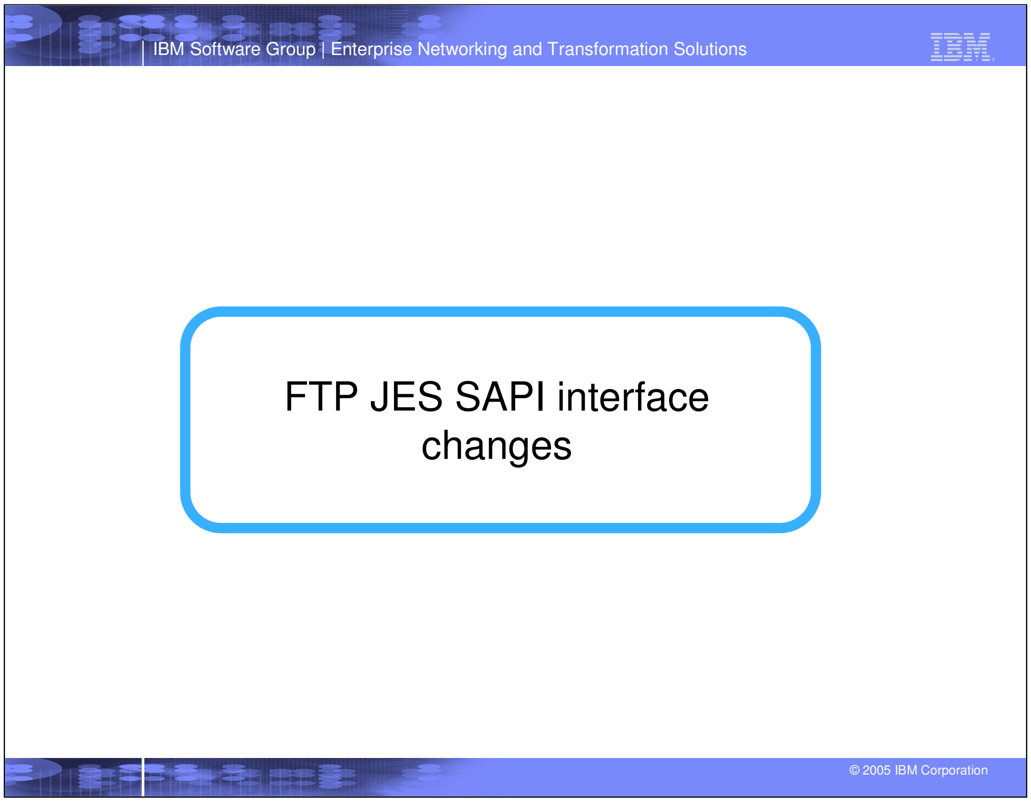
- **RACF in z/OS V1R7 supports mixed-case passwords**
  - Password possibilities increased
    - Still eight (8) characters
  - Likelihood of succeeding with a brute force attack on password combinations is reduced
- **z/OS FTP processes user-entered passwords in many scenarios and so far has always upper-case translated the passwords before handing them over the SAF interface.**
- **RACF administrator in z/OS V1R7 can toggle system-wide setting of password case support:**
  - SETROPTS PASSWORD(MIXED)
    - Passwords are used by FTP exactly as entered and handed over to SAF as entered
      - If stored password in RACF was last set when NOMIXED was active and never reset while MIXED was active, RACF will upper-case the passed value before checking for validity
  - SETROPTS PASSWORD(NOMIXED)
    - All passwords are upper-cased by FTP before handing over to SAF (as pre-V1R7)
- **FTP will in z/OS V1R7 adapt accordingly**
  - No new FTP configurations or options
- **You need to carefully evaluate FTP password usage before trying mixed-case passwords**
  - Remember the RACF options are system-wide and there are many other subsystems and applications to analyze before enabling mixed-case passwords



Things to think about

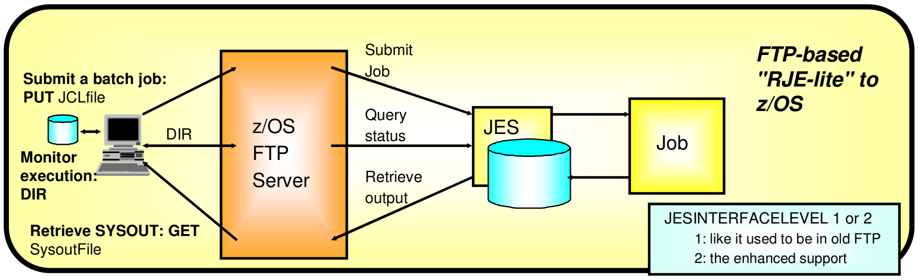
- **Interactive FTP client**
  - Educate interactive users!
  - Reply to password prompt with correct-case password!
- **FTP client using NETRC data set**
  - Code passwords in correct-case in NETRC data set
- **FTP client API programs**
  - Code password in correct case
- **FTP client batch jobs**
  - Code passwords in INPUT file in correct case
- **REXX programs stacking FTP client commands**
  - Code passwords in correct case in REXX programs
- **FTP server FTP.DATA ANONYMOUS statement**
  - If a password is coded on this statement, make sure it is coded in the correct case
  - Same consideration if you use the ANONYMOUS keyword on the EXEC PARM field when starting the FTP daemon - must be coded in correct case

For security reasons, FTP does not provide any traces that will print the password values. If RACF fails a login request because of mixed case problems, you will not be able to diagnose that using traditional FTP debugging technologies. For FTP client jobs, you can check all the locations above.



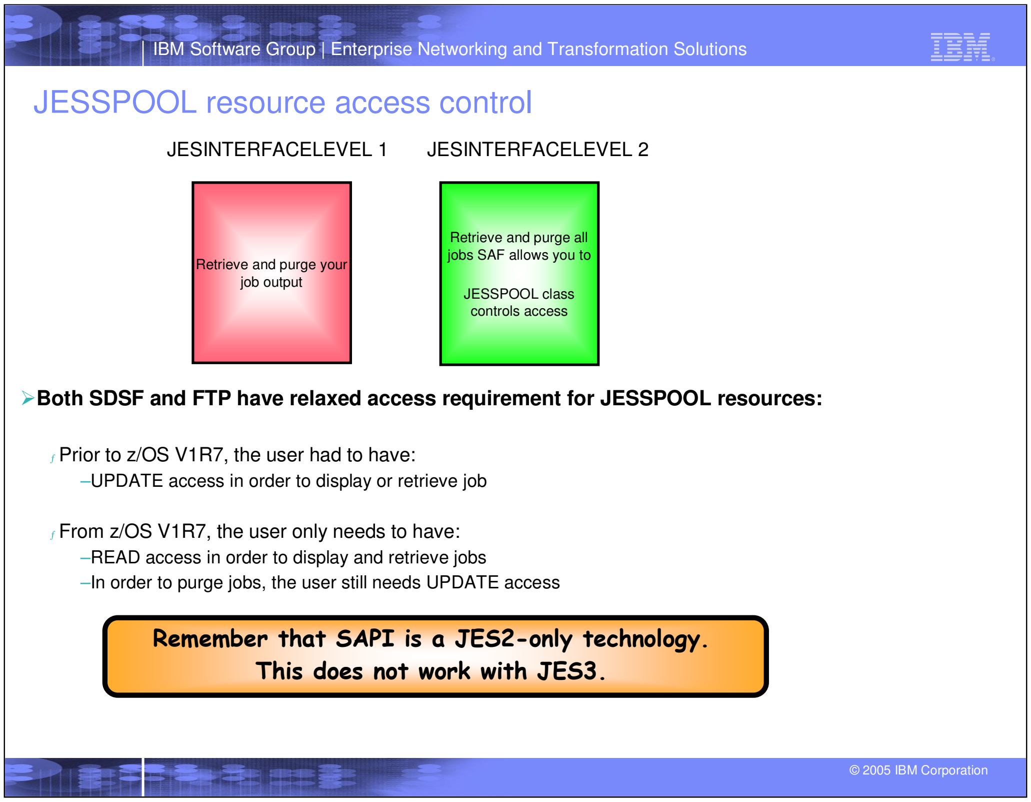
FTP JES SAPI interface changes

JES Interface Support in the z/OS FTP Server



Characteristics of JESINTERFACELEVEL 2 support:

- The JES interface uses the SAPI subsystem interface to JES2 and JES3
- All JES types (Jobs, started tasks, TSO, APPC)
- No jobname restrictions
- Details on DIR command output for jobs in input, active, or output status
- Filtering and access based on SAF interface - JESSPOOL and selected SDSF ISFCMD SAF profiles - allows access to jobs owned by others if proper SAF profiles are defined
- Filtering of jobs selected for DIR output is controlled via three SITE options:
  - JESJOBNAME (default <userID>*)
  - JESOWNER (default <userID>)
  - JESSTATUS (default ALL or OUTPUT or INPUT - determined by access to ISFCMD resources)
- ***Use of this interface (SAPI) prior to z/OS V1R7 required UPDATE access to JESSPOOL resources***

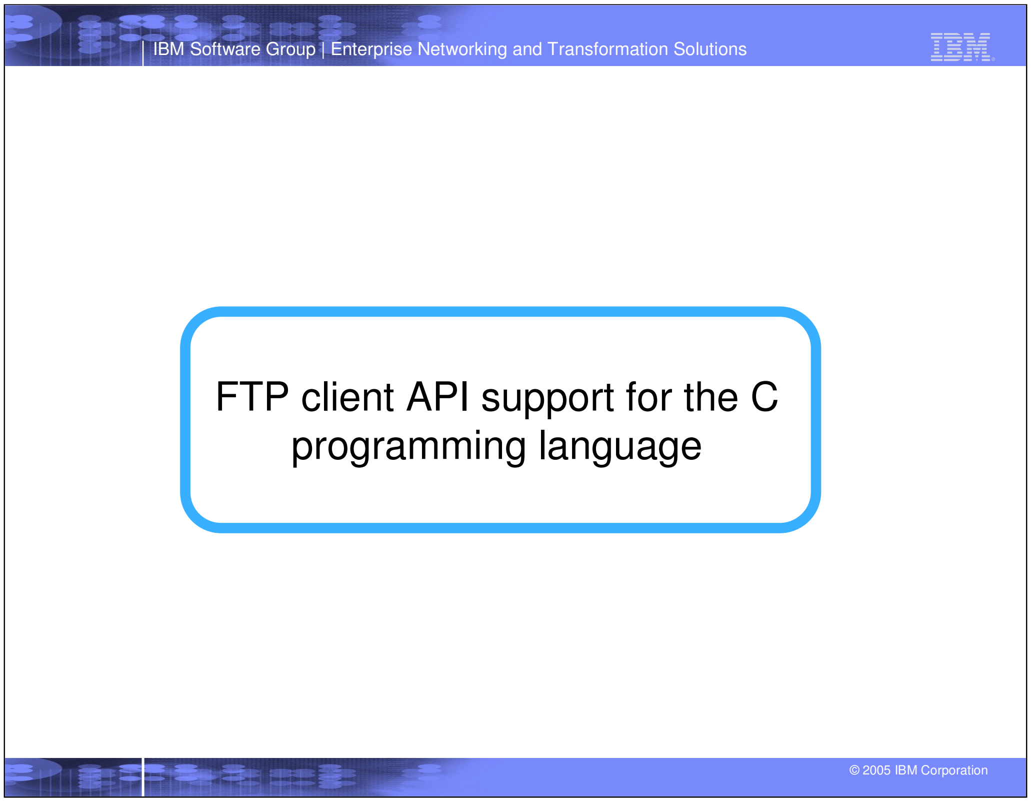


JESSPOOL resource access control

JESINTERFACELEVEL 1	JESINTERFACELEVEL 2
Retrieve and purge your job output	Retrieve and purge all jobs SAF allows you to JESSPOOL class controls access

- **Both SDSF and FTP have relaxed access requirement for JESSPOOL resources:**
  - Prior to z/OS V1R7, the user had to have:
    - UPDATE access in order to display or retrieve job
  - From z/OS V1R7, the user only needs to have:
    - READ access in order to display and retrieve jobs
    - In order to purge jobs, the user still needs UPDATE access

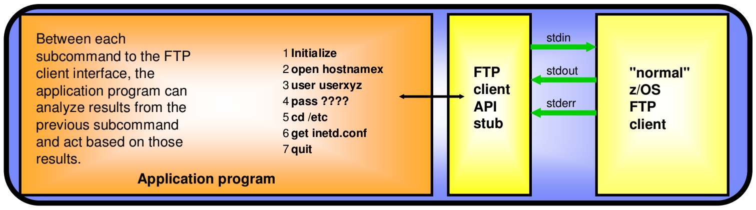
Remember that SAPI is a JES2-only technology.
This does not work with JES3.



FTP client API support for the C programming language

z/OS FTP client programming interface for improved automation and integration of z/OS file transfers

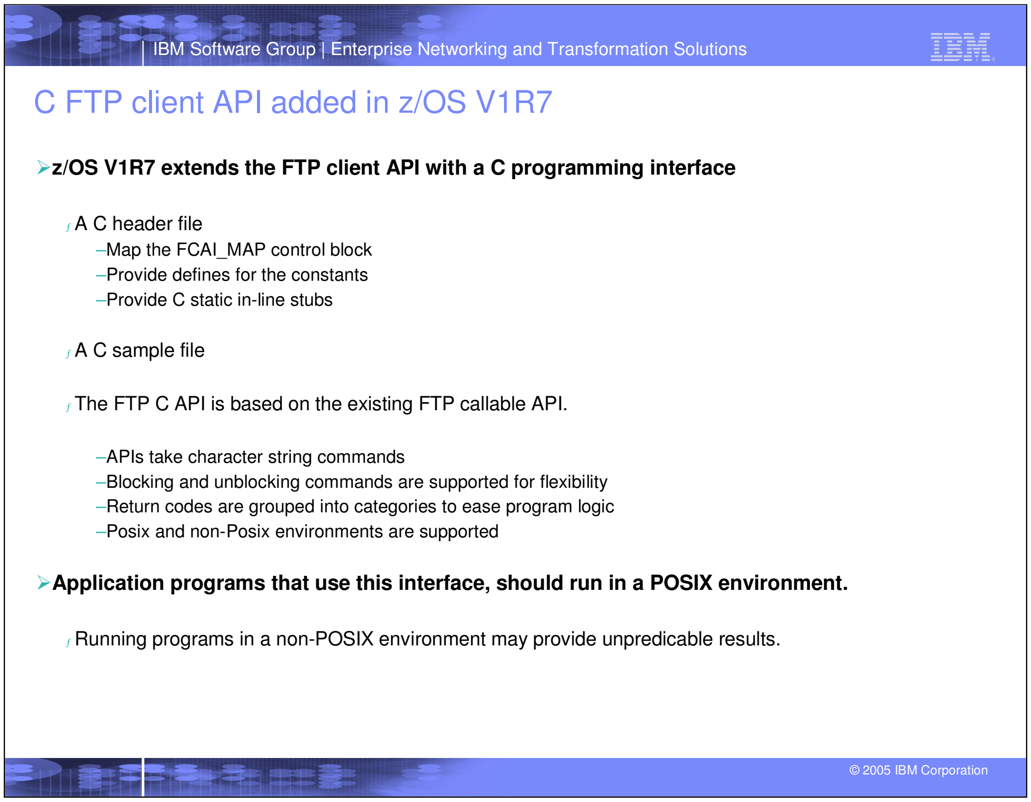
- Provides an interface that allows an application to programmatically invoke the FTP client on z/OS from common environments (UNIX shell, TSO, or MVS batch job)
- Characteristics of the interface:
  - z/OS V1R6 provides a callable interface to be used from assembler, Cobol, PL/I (or any z/OS supported programming language that supports a call interface)
    - ***z/OS V1R7 adds a C API version of this programming interface***
  - Interface is reentrant and does support multiple parallel FTP client sessions by tasks within an address space
  - For communication between the program and the interface, a simple set of commands and data areas are used (Mappings for common programming languages are provided)
  - Both blocking (wait for a response), and non-blocking (polling-mode) calls are supported
  - In non-blocking mode, progress notifications can be returned to the calling application as the transfer progresses
  - The simple commands tell the interface what to do, for example: initialize, terminate, execute an FTP client subcommand, process output from the FTP client subcommand that was executed, poll for command completion
  - Results are returned as structured fields in communication area control blocks (return codes from interface and server replies or possibly local subcommand) along with free-format replies from the FTP client code
  - Debugging options are provided



Between each subcommand to the FTP client interface, the application program can analyze results from the previous subcommand and act based on those results.

1 **Initialize**
2 **open hostnamex**
3 **user userxyz**
4 **pass ????**
5 **cd /etc**
6 **get inetd.conf**
7 **quit**

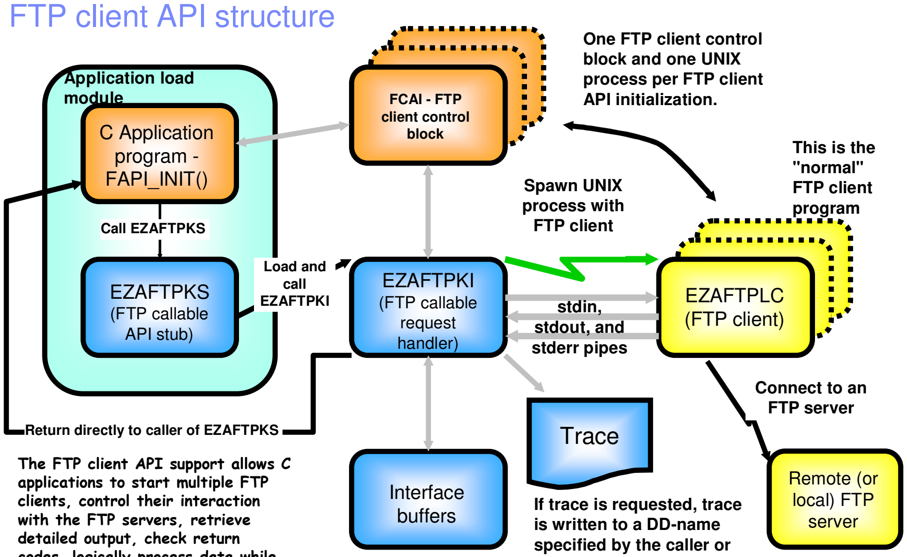
Application program



C FTP client API added in z/OS V1R7

- **z/OS V1R7 extends the FTP client API with a C programming interface**
  - A C header file
    - Map the FCAI_MAP control block
    - Provide defines for the constants
    - Provide C static in-line stubs
  - A C sample file
  - The FTP C API is based on the existing FTP callable API.
    - APIs take character string commands
    - Blocking and unblocking commands are supported for flexibility
    - Return codes are grouped into categories to ease program logic
    - Posix and non-Posix environments are supported
- **Application programs that use this interface, should run in a POSIX environment.**
  - Running programs in a non-POSIX environment may provide unpredicable results.

FTP client API structure

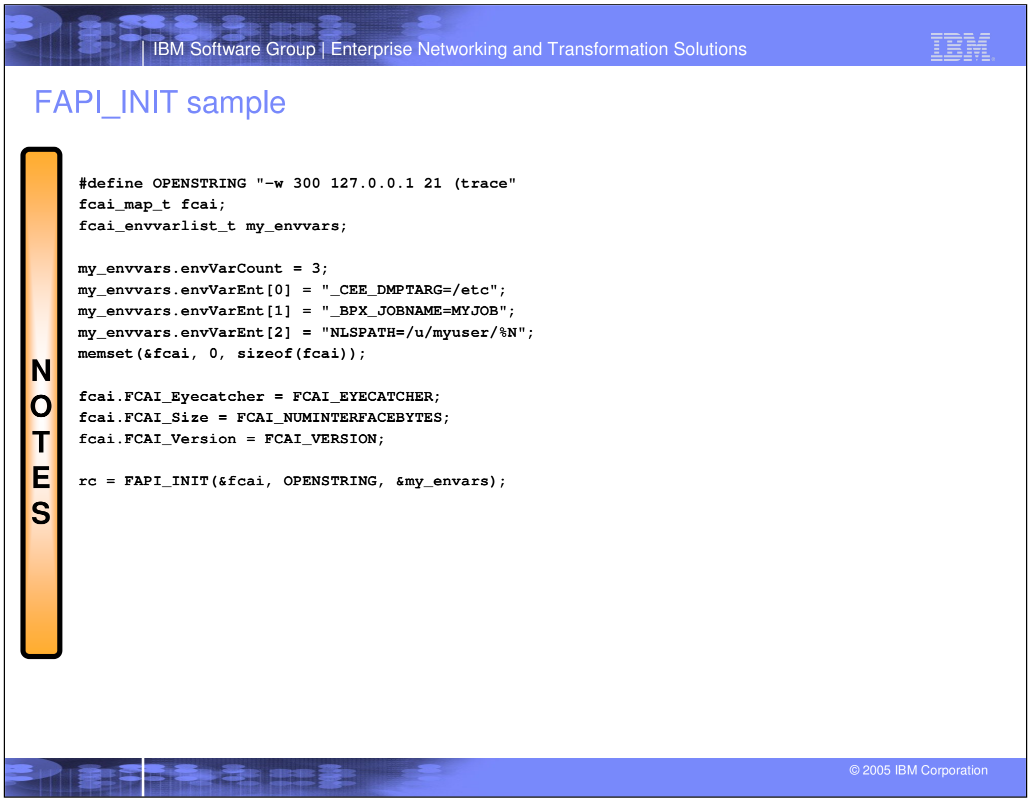

Application load module
C Application program - FAPI_INIT()
Call EZAFTPKS
EZAFTPKS (FTP callable API stub)
Load and call EZAFTPKI
FCAI - FTP client control block
EZAFTPKI (FTP callable request handler)
Spawn UNIX process with FTP client
stdin, stdout, and stderr pipes
EZAFTPLC (FTP client)
Connect to an FTP server
Remote (or local) FTP server
Trace
Interface buffers
Return directly to caller of EZAFTPKS

One FTP client control block and one UNIX process per FTP client API initialization.

This is the "normal" FTP client program

If trace is requested, trace is written to a DD-name specified by the caller or to a dynamically allocated SYSOUT file

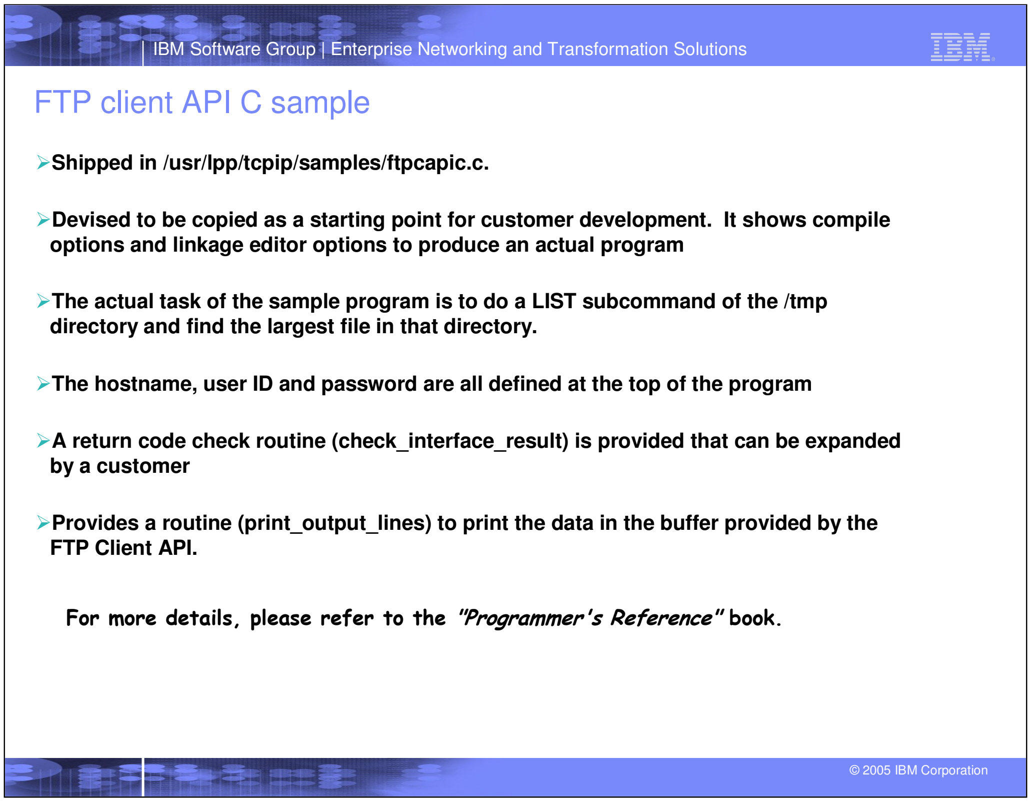
The FTP client API support allows C applications to start multiple FTP clients, control their interaction with the FTP servers, retrieve detailed output, check return codes, logically process data while transferring data, use blocking and non-blocking mode and much more!



FAPI_INIT sample

NOTES

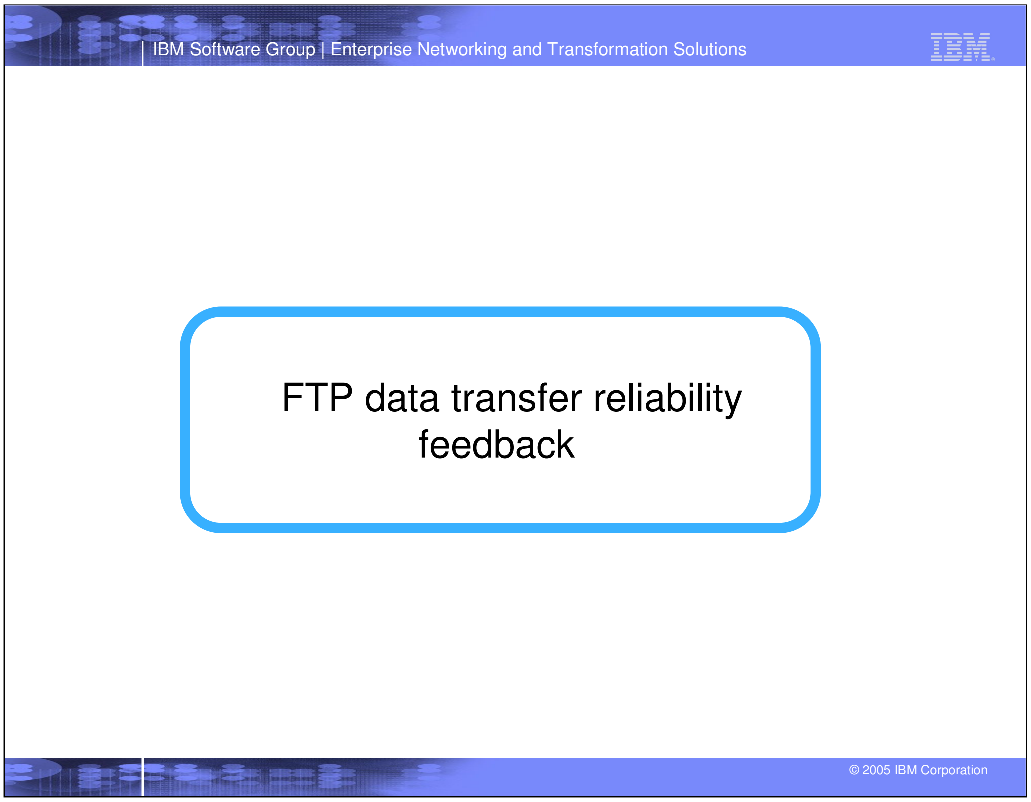
```
#define OPENSTRING "-w 300 127.0.0.1 21 (trace"
fcai_map_t fcai;
fcai_envvarlist_t my_envvars;

my_envvars.envVarCount = 3;
my_envvars.envVarEnt[0] = "_CEE_DMPTARG=/etc";
my_envvars.envVarEnt[1] = "_BPX_JOBNAME=MYJOB";
my_envvars.envVarEnt[2] = "NLSPATH=/u/myuser/%N";
memset(&fcai, 0, sizeof(fcai));

fcai.FCAI_Eyecatcher = FCAI_EYECATCHER;
fcai.FCAI_Size = FCAI_NUMINTERFACEBYTES;
fcai.FCAI_Version = FCAI_VERSION;

rc = FAPI_INIT(&fcai, OPENSTRING, &my_envars);
```



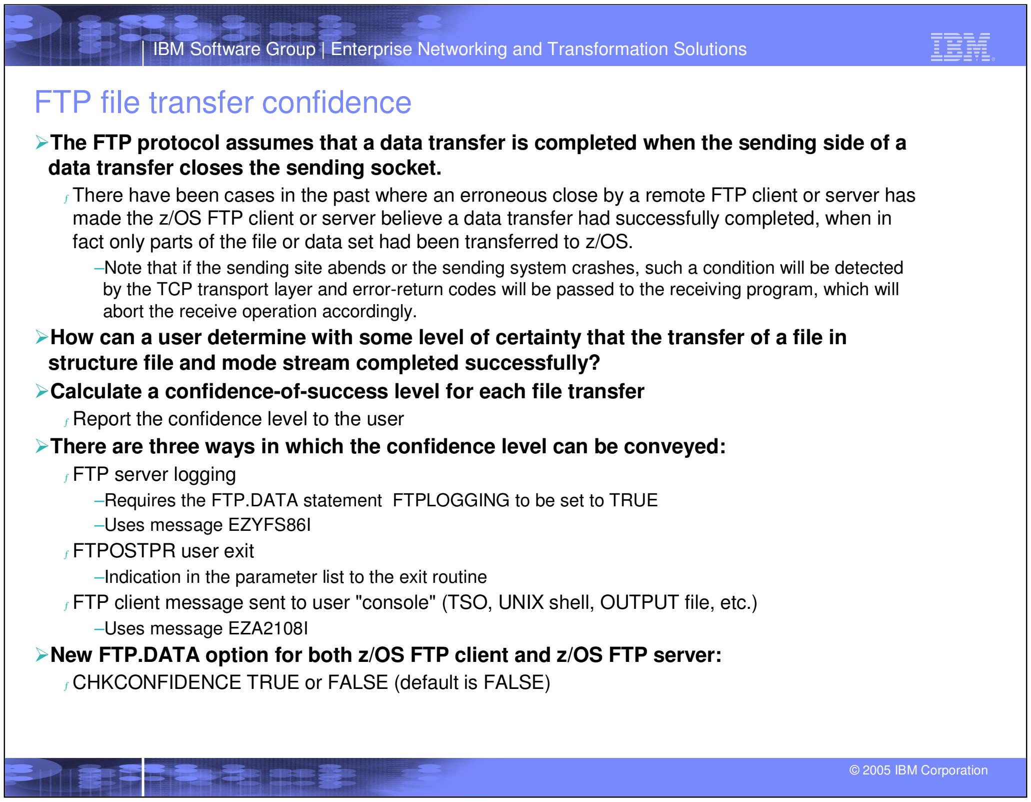
FTP client API C sample

- **Shipped in /usr/lpp/tcpip/samples/ftpcapic.c.**
- **Devised to be copied as a starting point for customer development. It shows compile options and linkage editor options to produce an actual program**
- **The actual task of the sample program is to do a LIST subcommand of the /tmp directory and find the largest file in that directory.**
- **The hostname, user ID and password are all defined at the top of the program**
- **A return code check routine (check_interface_result) is provided that can be expanded by a customer**
- **Provides a routine (print_output_lines) to print the data in the buffer provided by the FTP Client API.**

For more details, please refer to the *"Programmer's Reference"* book.

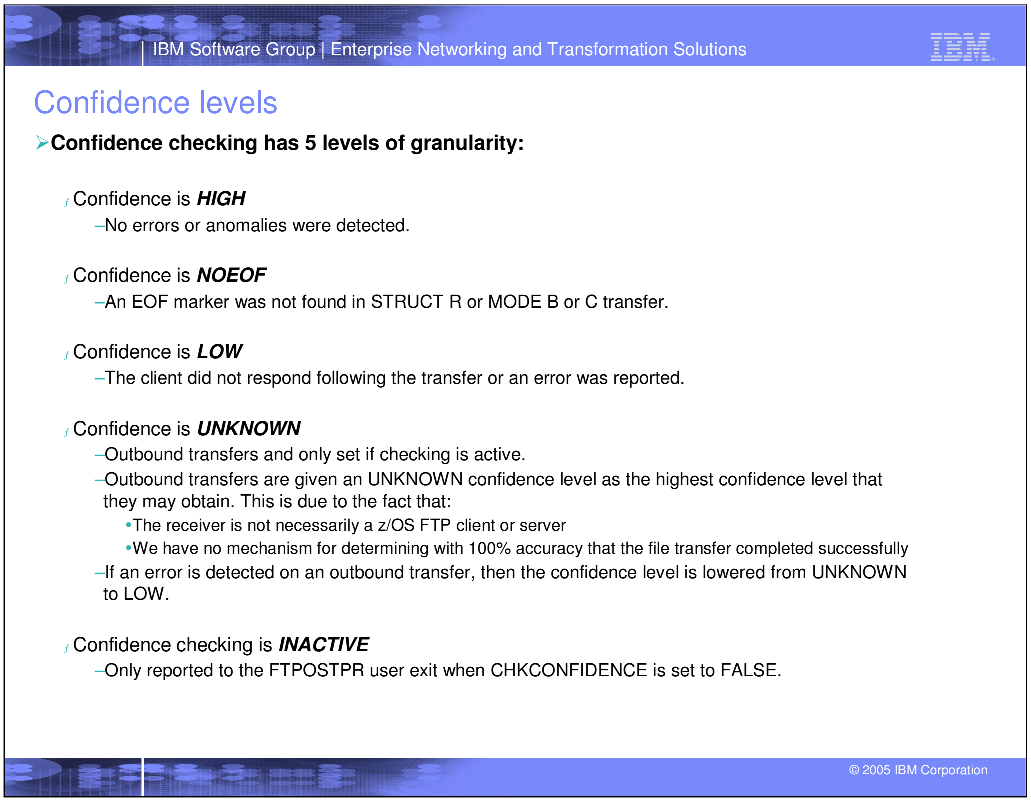


FTP data transfer reliability feedback



FTP file transfer confidence

- **The FTP protocol assumes that a data transfer is completed when the sending side of a data transfer closes the sending socket.**
  - There have been cases in the past where an erroneous close by a remote FTP client or server has made the z/OS FTP client or server believe a data transfer had successfully completed, when in fact only parts of the file or data set had been transferred to z/OS.
    - Note that if the sending site abends or the sending system crashes, such a condition will be detected by the TCP transport layer and error-return codes will be passed to the receiving program, which will abort the receive operation accordingly.
- **How can a user determine with some level of certainty that the transfer of a file in structure file and mode stream completed successfully?**
- **Calculate a confidence-of-success level for each file transfer**
  - Report the confidence level to the user
- **There are three ways in which the confidence level can be conveyed:**
  - FTP server logging
    - Requires the FTP.DATA statement FTPLOGGING to be set to TRUE
    - Uses message EZYFS86I
  - FTPOSTPR user exit
    - Indication in the parameter list to the exit routine
  - FTP client message sent to user "console" (TSO, UNIX shell, OUTPUT file, etc.)
    - Uses message EZA2108I
- **New FTP.DATA option for both z/OS FTP client and z/OS FTP server:**
  - CHKCONFIDENCE TRUE or FALSE (default is FALSE)



Confidence levels

- **Confidence checking has 5 levels of granularity:**

  - Confidence is ***HIGH***
    - No errors or anomalies were detected.
  - Confidence is ***NOEOF***
    - An EOF marker was not found in STRUCT R or MODE B or C transfer.
  - Confidence is ***LOW***
    - The client did not respond following the transfer or an error was reported.
  - Confidence is ***UNKNOWN***
    - Outbound transfers and only set if checking is active.
    - Outbound transfers are given an UNKNOWN confidence level as the highest confidence level that they may obtain. This is due to the fact that:
      - The receiver is not necessarily a z/OS FTP client or server
      - We have no mechanism for determining with 100% accuracy that the file transfer completed successfully
    - If an error is detected on an outbound transfer, then the confidence level is lowered from UNKNOWN to LOW.
  - Confidence checking is ***INACTIVE***
    - Only reported to the FTPOSTPR user exit when CHKCONFIDENCE is set to FALSE.

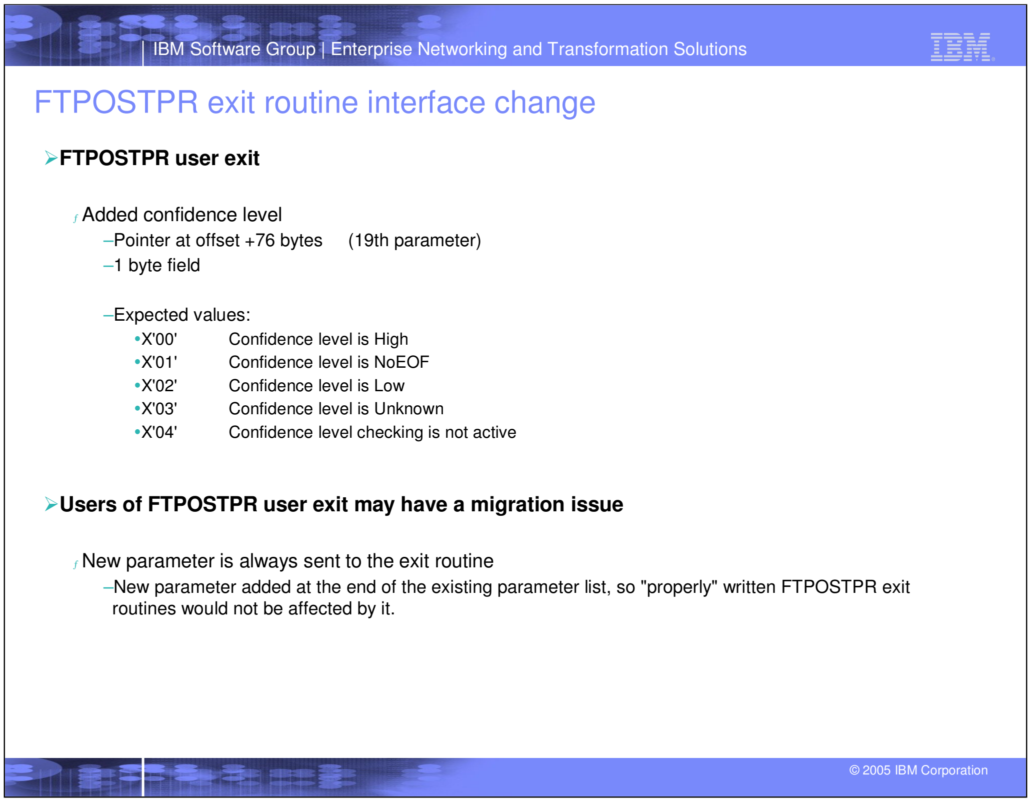
Reporting of confidence level

- **FTP server logging**
  - Confidence level is reported in syslog (such as /tmp/syslog.log)

```
Sep 22 15:20:41 MVS049 ftps[83886110]: EZYFS86I ID=FTPD100003 TRANS
Confidence=High
```

- **FTP client message**
  - Confidence level is reported to the client

```
get myfile /tmp/myfile
EZA1701I >>> EPSV
229 Entering Extended Passive Mode (|||1117|)
EZA1701I >>> RETR myfile
125 Sending data set USER1.MYFILE
250 Transfer completed successfully.
EZA2108I Confidence=High for GET of /tmp/myfile
EZA1617I 154 bytes transferred in 0.010 seconds.  Transfer rate 15.40 Kbytes/sec
```



FTPOSTPR exit routine interface change

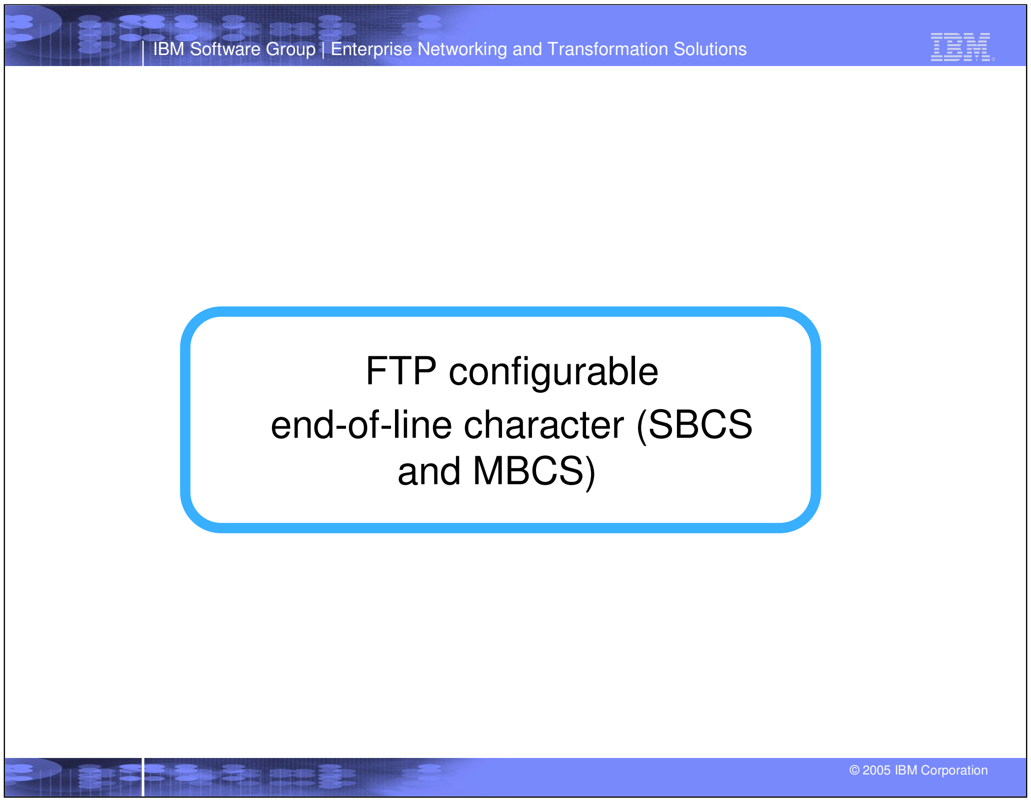
➢FTPOSTPR user exit

- Added confidence level
  - Pointer at offset +76 bytes (19th parameter)
  - 1 byte field
  - Expected values:

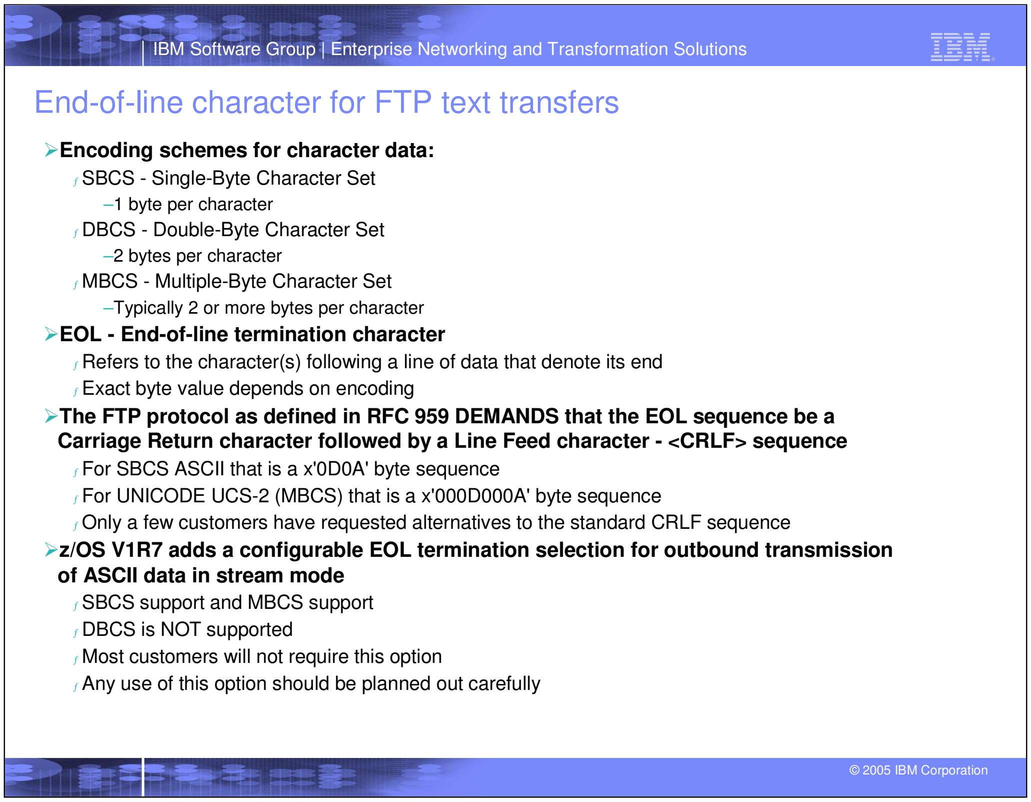
X'00'	Confidence level is High
X'01'	Confidence level is NoEOF
X'02'	Confidence level is Low
X'03'	Confidence level is Unknown
X'04'	Confidence level checking is not active

➢Users of FTPOSTPR user exit may have a migration issue

- New parameter is always sent to the exit routine
  - New parameter added at the end of the existing parameter list, so "properly" written FTPOSTPR exit routines would not be affected by it.

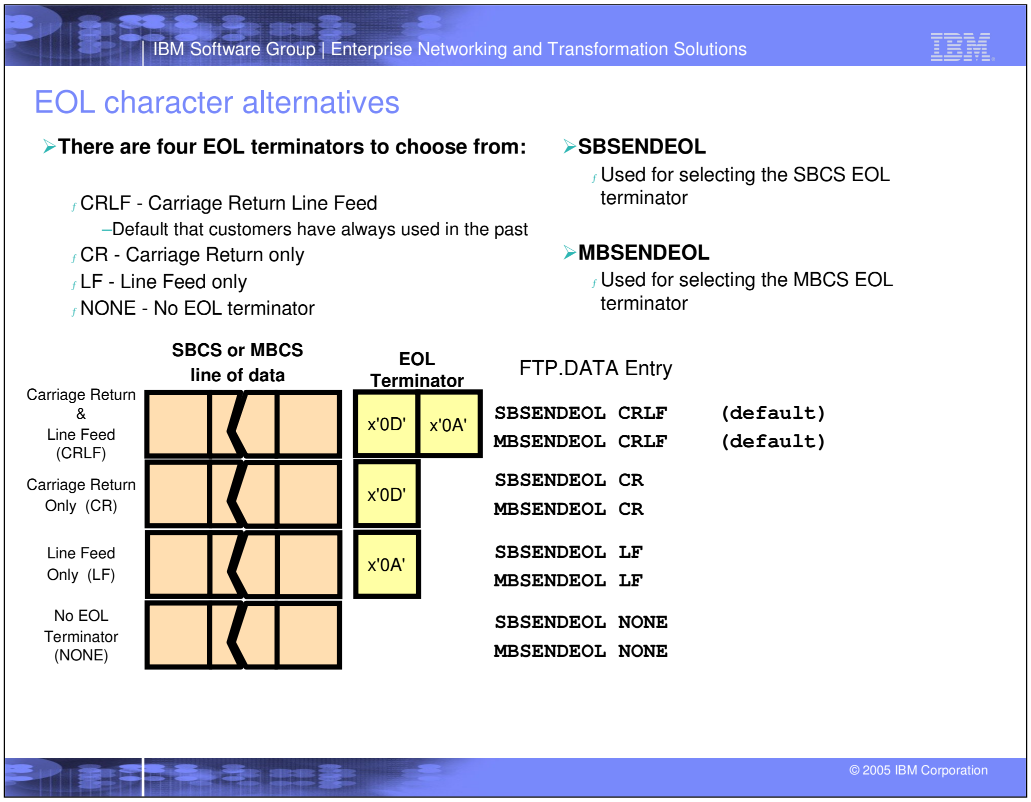


FTP configurable end-of-line character (SBCS and MBCS)



End-of-line character for FTP text transfers

- **Encoding schemes for character data:**
  - SBCS - Single-Byte Character Set
    - 1 byte per character
  - DBCS - Double-Byte Character Set
    - 2 bytes per character
  - MBCS - Multiple-Byte Character Set
    - Typically 2 or more bytes per character
- **EOL - End-of-line termination character**
  - Refers to the character(s) following a line of data that denote its end
  - Exact byte value depends on encoding
- **The FTP protocol as defined in RFC 959 DEMANDS that the EOL sequence be a Carriage Return character followed by a Line Feed character - <CRLF> sequence**
  - For SBCS ASCII that is a x'0D0A' byte sequence
  - For UNICODE UCS-2 (MBCS) that is a x'000D000A' byte sequence
  - Only a few customers have requested alternatives to the standard CRLF sequence
- **z/OS V1R7 adds a configurable EOL termination selection for outbound transmission of ASCII data in stream mode**
  - SBCS support and MBCS support
  - DBCS is NOT supported
  - Most customers will not require this option
  - Any use of this option should be planned out carefully



EOL character alternatives

- **There are four EOL terminators to choose from:**
  - CRLF - Carriage Return Line Feed
    - Default that customers have always used in the past
  - CR - Carriage Return only
  - LF - Line Feed only
  - NONE - No EOL terminator

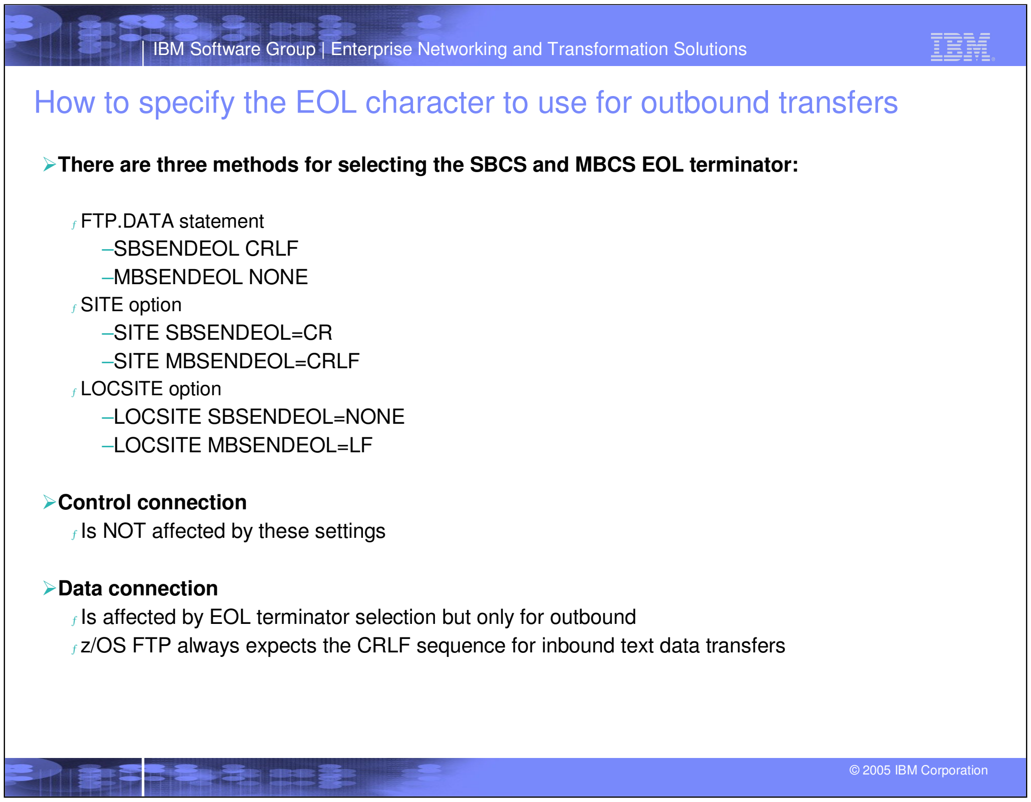
- **SBSENDEOL**
  - Used for selecting the SBCS EOL terminator

- **MBSENDEOL**
  - Used for selecting the MBCS EOL terminator

SBCS or MBCS line of data
EOL Terminator
Carriage Return & Line Feed (CRLF)
x'0D'
x'0A'
Carriage Return Only (CR)
x'0D'
Line Feed Only (LF)
x'0A'
No EOL Terminator (NONE)

FTP.DATA Entry

```
SBSENDEOL CRLF     (default)
MBSENDEOL CRLF     (default)

SBSENDEOL CR
MBSENDEOL CR

SBSENDEOL LF
MBSENDEOL LF

SBSENDEOL NONE
MBSENDEOL NONE
```

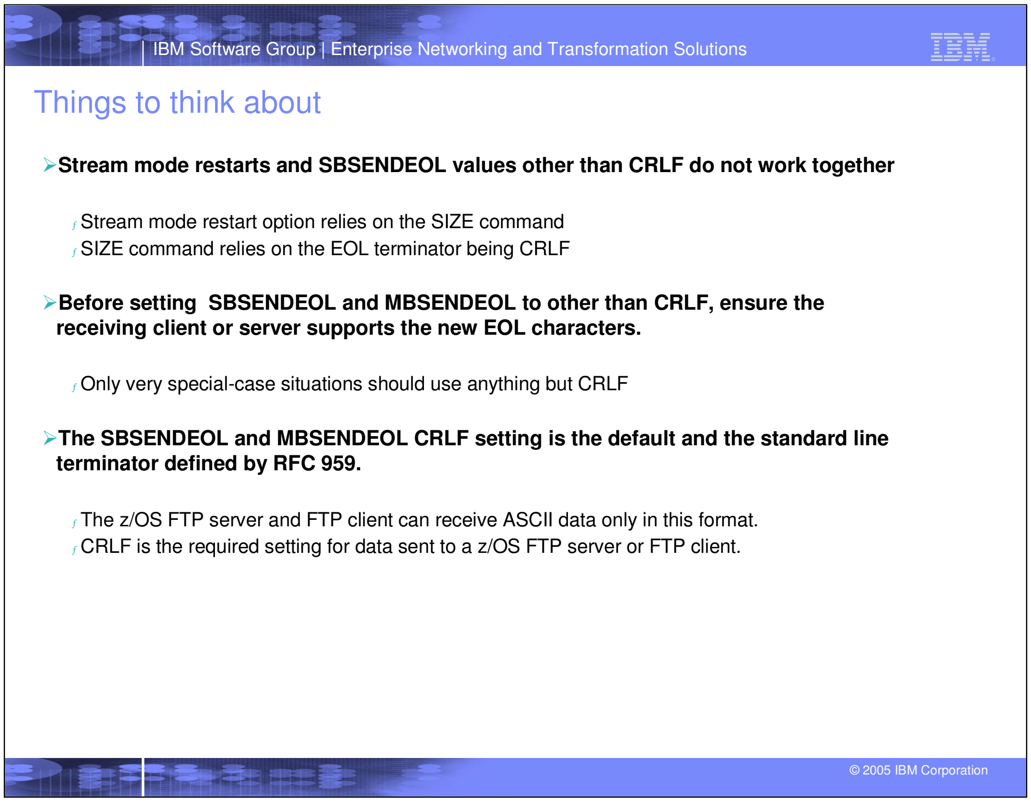


How to specify the EOL character to use for outbound transfers

- **There are three methods for selecting the SBCS and MBCS EOL terminator:**
  - FTP.DATA statement
    - SBSENDEOL CRLF
    - MBSENDEOL NONE
  - SITE option
    - SITE SBSENDEOL=CR
    - SITE MBSENDEOL=CRLF
  - LOCSITE option
    - LOCSITE SBSENDEOL=NONE
    - LOCSITE MBSENDEOL=LF

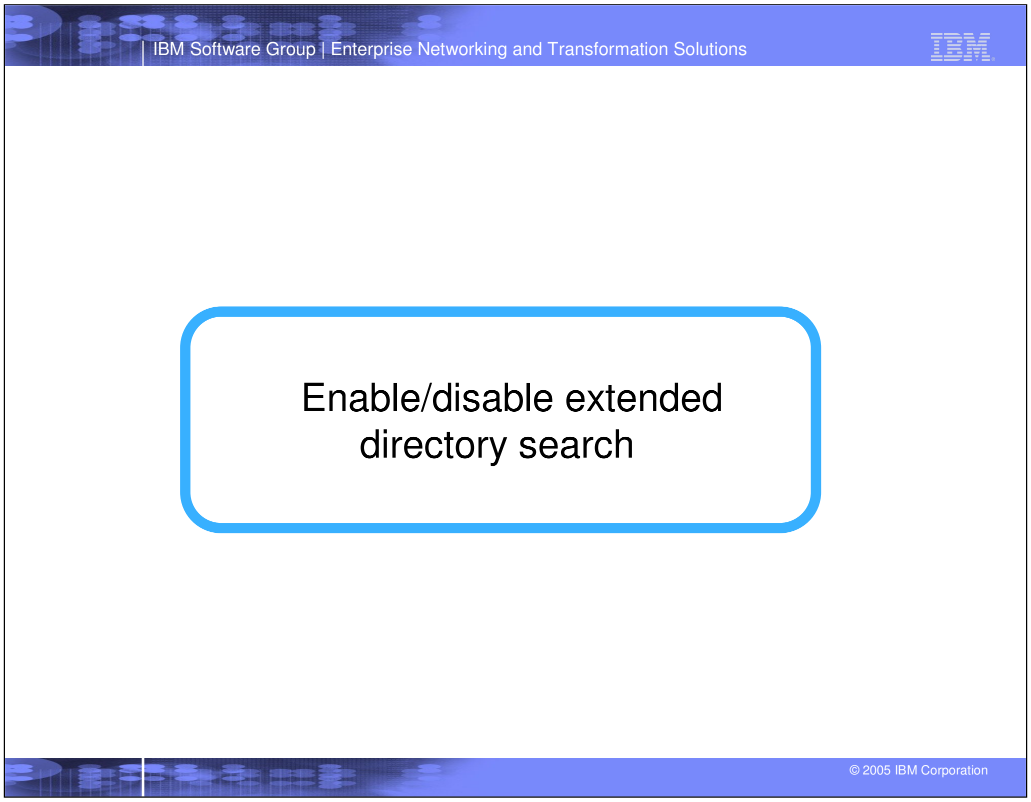
- **Control connection**
  - Is NOT affected by these settings

- **Data connection**
  - Is affected by EOL terminator selection but only for outbound
  - z/OS FTP always expects the CRLF sequence for inbound text data transfers

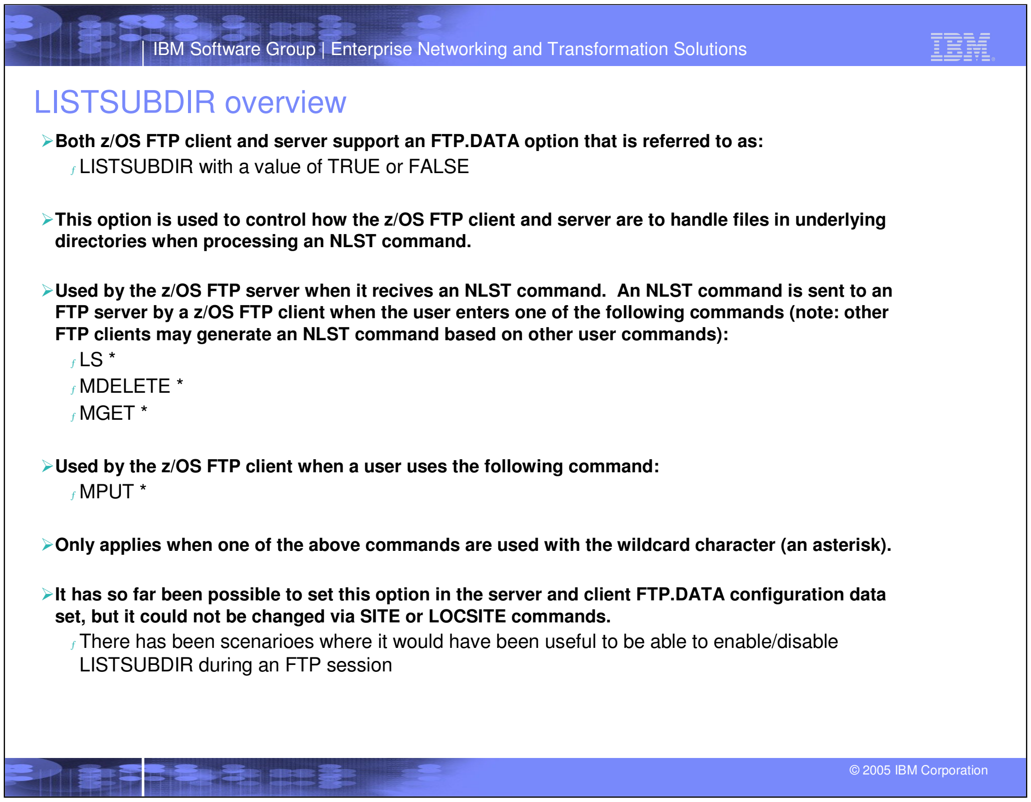


Things to think about

- **Stream mode restarts and SBSENDEOL values other than CRLF do not work together**
  - Stream mode restart option relies on the SIZE command
  - SIZE command relies on the EOL terminator being CRLF
- **Before setting SBSENDEOL and MBSENDEOL to other than CRLF, ensure the receiving client or server supports the new EOL characters.**
  - Only very special-case situations should use anything but CRLF
- **The SBSENDEOL and MBSENDEOL CRLF setting is the default and the standard line terminator defined by RFC 959.**
  - The z/OS FTP server and FTP client can receive ASCII data only in this format.
  - CRLF is the required setting for data sent to a z/OS FTP server or FTP client.



Enable/disable extended directory search

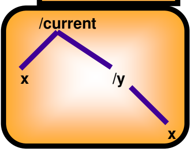


LISTSUBDIR overview

- **Both z/OS FTP client and server support an FTP.DATA option that is referred to as:**
  - LISTSUBDIR with a value of TRUE or FALSE
- **This option is used to control how the z/OS FTP client and server are to handle files in underlying directories when processing an NLST command.**
- **Used by the z/OS FTP server when it recives an NLST command. An NLST command is sent to an FTP server by a z/OS FTP client when the user enters one of the following commands (note: other FTP clients may generate an NLST command based on other user commands):**
  - LS *
  - MDELETE *
  - MGET *
- **Used by the z/OS FTP client when a user uses the following command:**
  - MPUT *
- **Only applies when one of the above commands are used with the wildcard character (an asterisk).**
- **It has so far been possible to set this option in the server and client FTP.DATA configuration data set, but it could not be changed via SITE or LOCSITE commands.**
  - There has been scenarioes where it would have been useful to be able to enable/disable LISTSUBDIR during an FTP session

An example of z/OS FTP client LISTSUBDIR usage

Directory structure on server



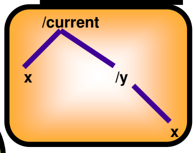
z/OS FTP client: LISTSUBDIR FALSE

```
lpwd
Local directory name set to hierarchical file /current
mput *
>>> PORT 127,0,0,1,4,11
200 Port request OK.
>>> STOR x
125 Storing data set /user1/x
250 Transfer completed successfully.
5 bytes transferred in 0.070 seconds.  Transfer rate 0.07
Kbytes/sec.
```

/current/x -> /user1/x

An example of z/OS FTP client LISTSUBDIR usage

Directory structure on server



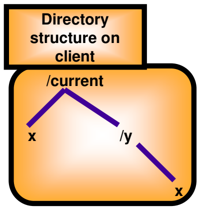
z/OS FTP client: LISTSUBDIR TRUE

```
lpwd
Local directory name set to hierarchical file /current
mput *
>>> PORT 127,0,0,1,4,11
200 Port request OK.
>>> STOR x
125 Storing data set /u/user1/x
250 Transfer completed successfully.
5 bytes transferred in 0.070 seconds.  Transfer rate 0.07
Kbytes/sec.
>>> PORT 127,0,0,1,4,12
200 Port request OK.
>>> STOR x
125 Storing data set /u/user1/x
250 Transfer completed successfully.
5 bytes transferred in 0.020 seconds.  Transfer rate 0.25
Kbytes/sec.
```

/current/x -> /user1/x

/current/y/x -> /user1/x - overwriting the file named x that was just transferred.

An example of z/OS FTP server LISTSUBDIR usage



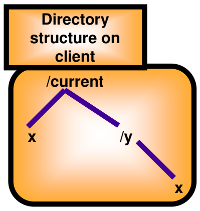
Directory structure on client

non-z/OS FTP client - z/OS FTP server with LISTSUBDIR FALSE

```
lpwd
Local directory name set to hierarchical file /current
mput *
>>> PORT 127,0,0,1,4,11
200 Port request OK.
>>> STOR x
125 Storing data set /user1/x
250 Transfer completed successfully.
5 bytes transferred in 0.070 seconds.  Transfer rate 0.07
Kbytes/sec.
```

/current/x -> /user1/x

An example of z/OS FTP server LISTSUBDIR usage



non-z/OS FTP client - z/OS FTP server with LISTSUBDIR TRUE

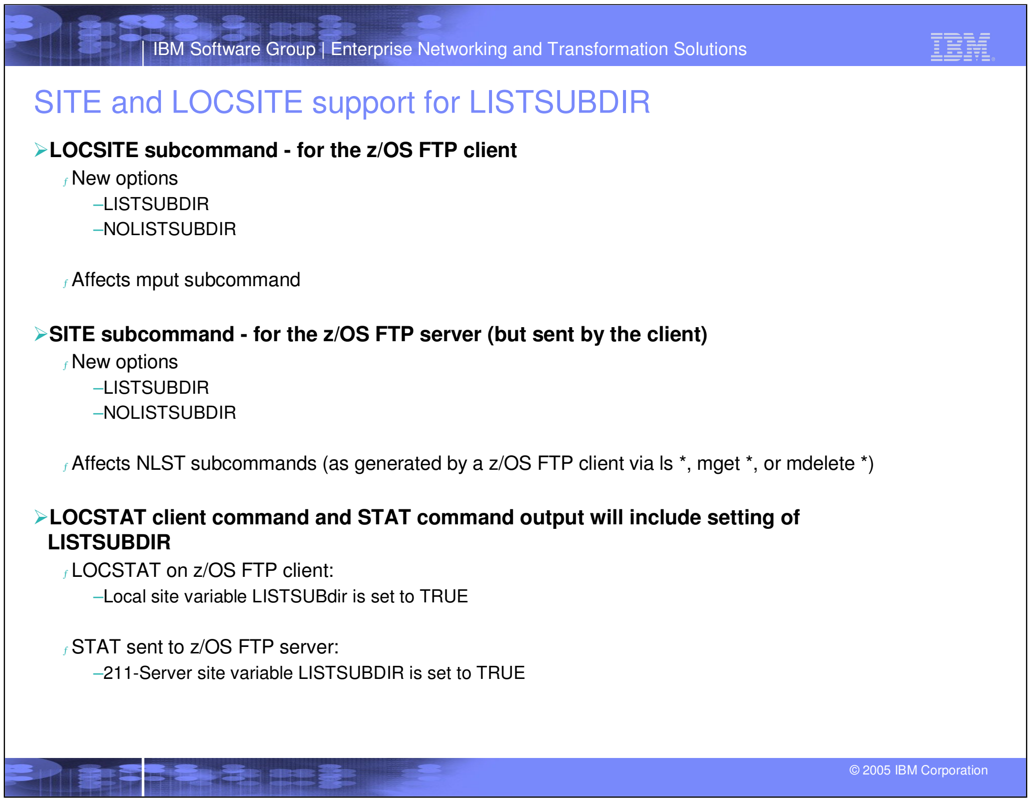
```
lpwd
Local directory name set to hierarchical file /current
mput *
>>> PORT 127,0,0,1,4,11
200 Port request OK.
>>> STOR x
125 Storing data set /u/user1/x
250 Transfer completed successfully.
5 bytes transferred in 0.070 seconds.  Transfer rate 0.07
Kbytes/sec.
>>> PORT 127,0,0,1,4,12
200 Port request OK.
>>> STOR x
125 Storing data set /u/user1/x
250 Transfer completed successfully.
5 bytes transferred in 0.020 seconds.  Transfer rate 0.25
Kbytes/sec.
```

If you had turned on the z/OS FTP client SUNIQUE option, the second file would have been stored as /u/user1/x1

SUNIQUE causes the client to use STOU instead of STOR commands.

/current/x -> /user1/x

/current/y/x -> /user1/x - overwriting the file named x that was just transferred from the /current directory



SITE and LOCSITE support for LISTSUBDIR

➢**LOCSITE subcommand - for the z/OS FTP client**

ƒ New options

–LISTSUBDIR

–NOLISTSUBDIR

ƒ Affects mput subcommand

➢**SITE subcommand - for the z/OS FTP server (but sent by the client)**

ƒ New options

–LISTSUBDIR

–NOLISTSUBDIR

ƒ Affects NLST subcommands (as generated by a z/OS FTP client via ls *, mget *, or mdelete *)

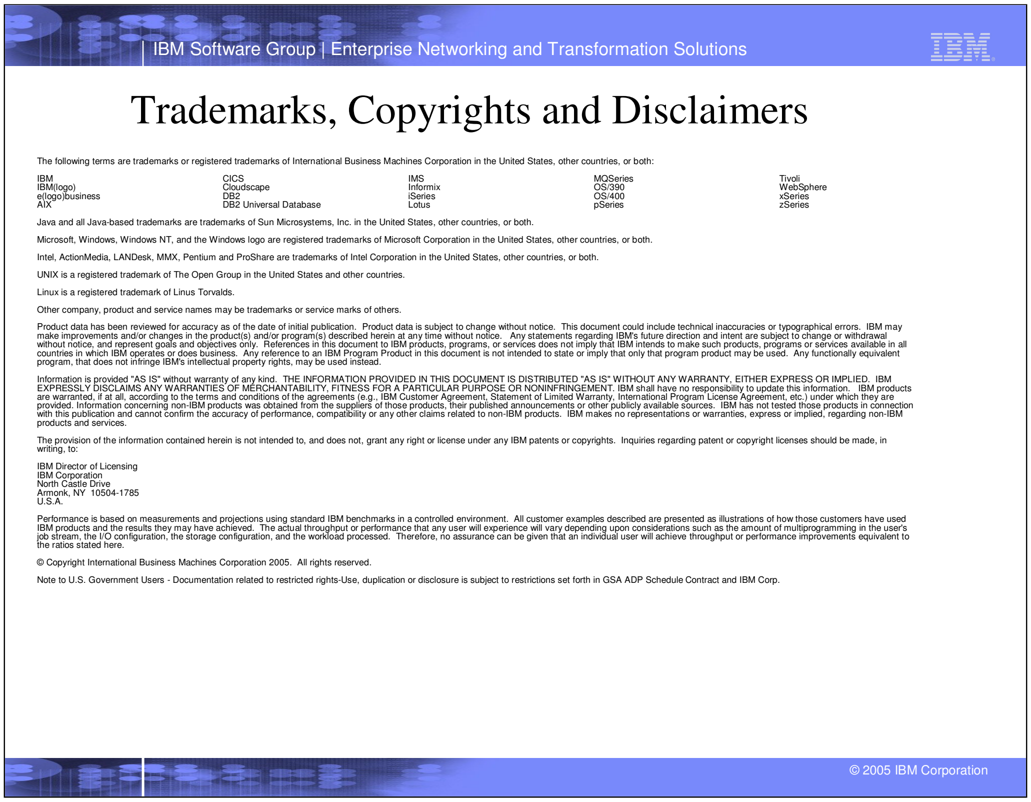
➢**LOCSTAT client command and STAT command output will include setting of LISTSUBDIR**

ƒ LOCSTAT on z/OS FTP client:

–Local site variable LISTSUBdir is set to TRUE

ƒ STAT sent to z/OS FTP server:

–211-Server site variable LISTSUBDIR is set to TRUE



Trademarks, Copyrights and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
IBM(logo)
e(logo)business
AIX

CICS
Cloudscape
DB2
DB2 Universal Database

IMS
Informix
iSeries
Lotus

MQSeries
OS/390
OS/400
pSeries

Tivoli
WebSphere
xSeries
zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.