How can I validate whether files were successfully transferred to EFT (integrity validation)?

THE INFORMATION IN THIS ARTICLE APPLIES TO:

- EFT (All Versions)

QUESTION

How can I validate whether files were successfully transferred to EFT (integrity validation)?

ANSWER

Although TCP/IP ensures that all packets are received properly, mishandling by the application can occur, leading the client to believe that a transfer was successful when it was not.

The Server's file integrity command is defined as *XCRC*. Once an XCRC-enabled client (such as CuteFTP) performs a transfer, it can request the Server to do a checksum calculation on the file. If it matches the checksum on the client, then the transfer is deemed successful. Performing XCRC checksum calculations is processor intensive; enable or disable the feature accordingly.

XCRC is a proprietary command and is not defined nor endorsed by any FTP-related RFC. Competing servers who want to implement this command may do so using the syntax described below.

XCRC <File Name>

XCRC <File Name>, <EP>

XCRC <File Name>, <SP>, <EP>

SP = Starting Point in bytes (from where to start CRC calculating)

EP = Ending Point in bytes (where to stop CRC calculating)

FTP Client Log Example

```
COMMAND:> XCRC "/Program Files/MSN Gaming Zone/Windows/chkrzm.exe" 0 42575
```

- SP and EP are optional parameters. If not specified then it calculates the CRC for the whole file. If only EP is specified, then the CRC calculation starts from the beginning of the file to the EP.
- This command can be used for a single file at a time. It does not allow file lists as parameters.
- The standard CRC32 algorithm is used (for speed and efficiency).
- A client can invoke this command for uploads, downloads, and single and Multi-Part Transfers.

Server Reply	**Indicates**
250 <XCRC>	calculated CRC value
450 Requested file action not taken	file is busy
550 Requested action not taken	file is not found or has no read permission; or the SP or EP are not correct

File Integrity Checking in FTP and SFTP

FTP provides a very raw mechanism to transfer files – the data for a file is uploaded/downloaded as a stream of bits over a TCP/IP connection. There is no additional overhead on that operation, so it is fast; however, this also means that there are no intrinsic mechanisms for ensuring that the file got to the other end intact. That is why many servers, including EFT Server, support the XCRC command, an extension to the FTP protocol. The XCRC command performs a CRC32 checksum over the file (either the whole file, or a portion of the file if a byte range is specified on the command line). This gives a 32-bit value that the Server computes on the file, which can be compared to the 32-bit value computed on

the client side. If both files are the same size and the CRC32 matches, then there is a very high probability that the files are identical and files were transferred correctly.

In contrast, SFTP ***does*** add overhead to the transfer of files. As a file is transferred between client and server, it is broken up into smaller chunks called "packets." For example, suppose each packet is 32KB. The SFTP protocol does a checksum on each 32KB packet as it is sent, and includes that checksum along with that packet. The receiver gets that packet and decrypts the data, and then verifies the checksum. The checksum itself is "stronger" than the CRC32 checksum (because SFTP uses a 128-bit or higher checksum, such as MD5 or SHA, and because this is done on each and every packet, there is a very granular integrity checking that is accomplished as part of the transfer. Thus, the protocol itself is slower (because of the additional overhead), but the successful completion of a transfer means, de facto, that it has be transferred integrally and there is no need for an additional check.

If you want to verify integrity, then use SFTP protocol and it is built-in; or, use FTP and after a transfer issue the raw command "XCRC /path/to/file" and read the results, comparing to a locally calculated CRC32 on the client side. EFT Server also supports querying the CRC32 value over the HTTP/S protocol by issuing a "HEAD" request to the file. The result of a HEAD method invocation on the HTTP/S engine will result in a response that includes the "X-CRC" header, which contains the CRC32 value of the file in question. This can be compared to the CRC32 computed over the local file, just like in the FTP case.