EXHIBIT 2064

Part 2 of 2

27 Segmentation and reassembly

27.1 Introduction

Segmentation and reassembly are the FC-2 functions provided to subdivide application data corresponding to an Information Category to be transferred into Payloads, compute Relative Offset values, embed each Payload and the corresponding Relative Offset value in an individual frame, transfer these frames over the link, and reassemble at the receiving end the application data of the Information Category as sent by the sending end.

Application data mapping
Mapping application data to Upper Level Protocols (ULPs) is outside the scope of FC-PH. ULPs maintain the status of application data transferred.

27.2 Sending end

An upper level (a level above FC-2) at the sending end shall define a Relative Offset space for each Information Category for which Relative Offset usage is supported. An upper level specifies a block or a collection of subblocks to be transferred within a Sequence.

27.2.1 Relative Offset space

The Relative Offset space shall start from zero, representing an upper level-defined origin, and extend to its highest value. Application data for an Information Category transferred may be mapped to all or portions of Relative Offset space.

27.2.2 Block

A block is an upper level construct of application data related to a single Information Category and transferred within a single Sequence. If Continuously Increasing Relative Offset is used, a block shall be specified. A block is allowed to map to all or part of Relative Offset space of the Information Category. An upper level shall specify an Initial Relative Offset (IRO_i) for each block. The Initial Relative Offset value is allowed to be zero or non-zero.

27.2.3 Subblock

A subblock is an upper level construct which contains partial data for a single Information Category. A collection of subblocks is specified for a given Information Category to be transferred within a Sequence. The subblocks shall be specified, only if the Sequence Recipient supports Random Relative Offset. An upper level shall specify an Initial Relative Offset for each subblock. The Initial Relative Offset value is allowed to be zero or non-zero. The Initial Relative Offset values of subblocks of a given Information Category transmitted consecutively are allowed to be random.

The collection of subblocks for a single Information Category is allowed to map to portions or all of Relative Offset space for the Information Category.

27.2.4 Sequence

The blocks to be transferred within a single Sequence and the Information Category for each block shall be specified to FC-2 by an upper level. The collections of subblocks to be transferred within a Sequence and the information Category for each subblock shall be specified to FC-2 by an upper level.

27.2.5 Relationship between Sequences

The Relative Offset relationship between multiple Sequences of a given Information Category shall be specified by an upper level. This relationship is transparent to FC-2.

27.3 FC-2

Mechanisms
FC-2 mechanisms to support this function are

a) Relative Offset or SEQ_CNT
b) Sequence
c) F_CTL bit indicating the presence of Relative Offset in the parameter field (see 18.2)

Relative Offset
The Parameter field in the Frame_Header may be used to hold the value of Relative Offset

Payload related to an upper level-defined origin for a given Information Category.

Sequence Count (SEQ_CNT)
If Relative Offset is not used for reassembly, the Information Category shall be reassembled using the SEQ_CNT. The reassembly shall be performed by joining or concatenating the Payloads of the Data frames in the continuously increasing order of SEQ_CNTs starting from the SEQ_CNT of the first Data frame to that of the last Data frame of a given Sequence.

For an Information Category i within a given Sequence, the reassembly is expressed as
Application $data_i$ =
PL_i (m) || PL_i (m+1) || ooo || PL_i (m+n)
where PL (m) represents Payload
of a frame with SEQ_CNT m
n is the total frame count
within the Sequence
m is the lowest SEQ_CNT (zero or non-zero) and m+n is the highest SEQ_CNT.

If the SEQ_CNT wraps to zero from hex FFFF within a Sequence, the reassembly shall be continued according to modulo 65536 arithmetic (i.e., SEQ_CNT = 0 follows SEQ_CNT = hex FFFF).

27.4 Login

The following Common Service Parameters related to segmentation and reassembly are interchanged during N_Port Login (see figure 61):

a) Relative Offset by Information Category
b) Continuously Increasing or Random Relative Offset

Through the interchange of these Login parameters, the Sequence Recipient indicates its Relative Offset requirements to the Sequence Initiator. The Sequence Recipient indicates Relative Offset support or non-support for each Information Category. For the Relative Offset supported Information Categories, the Sequence Recipient collectively indicates Continuously Increasing or Random Relative Offset requirement.

The Sequence Initiator shall follow the Relative Offset requirements of the Sequence Recipient, for Information Categories supported and not supported.

27.5 Segmentation rules summary

Segmentation summary rules are listed and referred to table 110.

a) The Sequence Initiator shall be responsible for segmentation. The Sequence Initiator shall follow the Relative Offset requirements of the Sequence Recipient for Information Categories.

b) An upper level at the sending end shall specify to the sending FC-2 one or more blocks to be transferred as a Sequence, the Information Category for each block, an Relative Offset space, and the Initial Relative Offset for each Information Category. The Initial Relative Offset value may be zero or non-zero.

c) The Sequence Initiator shall use the specified Relative Offset space for each Information Category and transfer one or more blocks specified in a single Sequence.

d) If the Sequence Recipient does not support Relative Offset for one or more Information Categories, the Sequence Initiator shall transmit each of these Information Categories as a contiguous block. The Sequence Initiator shall set the Relative Offset present bit in F_CTL to zero, indicating that the parameter field is not meaningful.

e) If the Sequence Recipient supports Relative Offset for one or more Information Categories and has specified during Login this support as Continuously Increasing Relative Offset, the Sequence Initiator shall transmit each of these Information Categories with Continuously Increasing Relative Offset.

1) The Sequence Initiator shall set the Relative Offset present F_CTL bit to one.
2) The Sequence Initiator shall use the Initial Relative Offset specified by the upper level for the Relative Offset RO_i in the first frame of the block, namely,
$RO_i(0)$ = Initial Relative Offset (IRO_i) for the Information Category i
3) The Sequence Initiator shall use for all other frames of the block, the Relative Offset computed as follows:
$RO_i(n+1)$ = $RO_i(n)$ + Length of $Payload_i(n)$
where n is ≥ 0 (zero) and represents the consecutive frame count of frames for a

given Information Category within a single Sequence.

4) Above steps 1 through 3 shall be repeated for each block within the Sequence.

f) If the Sequence Recipient supports Relative Offset for one or more Information Categories and has specified during Login this support as Random Relative Offset, the Sequence Initiator is permitted to transmit each of these Information Categories with Random Relative Offset.

1) The Sequence Initiator shall set the Relative Offset present F_CTL bit to one.

2) The Sequence Initiator shall use the Initial Relative Offset specified by the upper level for the Relative Offset RO_i in the first frame of the subblock, namely,
$RO_i(0)$ = Initial Relative Offset (IRO_i) for the Information Category i

3) The Sequence Initiator shall use for all other frames of the subblock, the Relative Offset computed as follows:
$RO_i(n+1) = RO_i(n) +$ Length of $Payload_i(n)$
where n is ≥ 0 (zero) and represents the consecutive frame count of frames for the subblock for a given Information Category within a single Sequence.

4) Above steps 1 through 3 shall be repeated for each subblock within the Sequence.

27.6 Reassembly rules summary

Reassembly summary rules are listed and referred to table 110.

a) The Sequence Recipient shall be responsible for reassembly of each Information Category received within the Sequence. The Sequence Recipient shall use Relative Offset or SEQ_CNT field, as specified, to perform the reassembly and make the blocks available to the receiving upper level as sent by the sending upper level.

b) The Sequence Recipient shall reassemble each Information Category within its Relative Offset space specified by the sending upper level.

c) If the Sequence Recipient has specified during Login non-support of Relative Offset for one or more Information Categories, the Sequence Recipient shall verify Relative Offset present F_CTL bit for zero value and reassemble each of these Information Categories using SEQ_CNT.

If this F_CTL bit is set to one for any of these Information Categories, the Sequence Recipient shall issue a P_RJT with a reason code of "Relative Offset not supported."

d) If the Sequence Recipient has indicated during Login Relative Offset support for one or more Information Categories and specified this support as Continuously Increasing Relative Offset, the Sequence Recipient shall verify Relative Offset present F_CTL bit for one and assemble each of these Information Categories using Relative Offset.

If the Sequence Initiator lacks the Relative Offset capability and has set the bit to zero, the Sequence Recipient shall use SEQ_CNT for reassembly.

e) If the Sequence Recipient has indicated during Login Relative Offset support for one or more Information Categories and specified this support as Random Relative Offset, the Sequence Recipient shall verify Relative Offset present F_CTL bit for one and assemble each of these Information Categories using Relative Offset.

If the Sequence Initiator lacks the Relative Offset capability and has set the bit to zero, the Sequence Recipient shall use SEQ_CNT for reassembly.

f) If the Sequence Recipient supports Continuously Increasing Relative Offset and detects random Relative Offsets, the Sequence Recipient shall issue P_RJT with the reason code of "Relative Offset out of bounds".

Table 110 - Segmentation and reassembly rules summary

Case	Relative Offset support by Sequence Recipient	Sequence Initiator action (Segmentation)	Sequence Recipient action (Reassembly)
1	Not supported	- F_CTL Relative Offset present bit = 0 - Parameter field not meaningful	- Relative Offset shall not be used (ignore parameter field) - SEQ_CNT shall be used
		- F_CTL Relative Offset present bit = 1 - Parameter field = Relative Offset	- Issue P_RJT - Reason code = Relative Offset not supported
2	Continuously Increasing Relative Offset supported	- F_CTL Relative Offset present bit = 1 - Parameter field = Relative Offset - First frame of a block: $RO_i(0) = IRO_i$ for the block specified - All other frames of the block: $RO_i(n+1) = RO_i(n) +$ Length of $Payload_i(n)$	Relative Offset shall be used
		- F_CTL Relative Offset present bit = 0 - Parameter field not meaningful	- Ignore parameter field - SEQ_CNT shall be used
3	Random Relative Offset supported	- F_CTL Relative Offset present bit = 1 - Parameter field = Relative Offset - Initial Relative Offset for subblocks permitted to be random - First frame of a subblock: $RO_i(0) = IRO_i$ for the subblock specified - All other frames of the subblock: $RO_i(n+1) = RO_i(n) +$ Length of $Payload_i(n)$	Relative Offset shall be used
		- F_CTL Relative Offset present bit = 0 - Parameter field not meaningful	- Ignore parameter field - SEQ_CNT shall be used

Note:

If RO value in the Parameter field is out of bounds, the Sequence Recipient shall issue a P_RJT with a reason code of "Invalid Parameter field".

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.