

AS/NZS 3646.4 Supp 1:1995

ISO 8571-4:1988/Amd.2:1993

AS/NZS 3646.4
Supplement 1:1995

**Information processing systems—
Open Systems Interconnection—
File transfer, access and
management**

**Part 4: File protocol specification—
Overlapped access**

**(Supplement to
AS/NZS 3646.4:1993)**

[ISO title: Information processing systems—Open Systems
Interconnection—File Transfer, Access and Management
Part 4: File Protocol Specification, Amendment 2: Overlapped access]

STANDARDS AUSTRALIA ✓

STANDARDS NEW ZEALAND
Documents



Document Sharing Hub

AS/NZS 3646.4 Supp 1:1995

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT/1, Information Systems—Interconnection. It was approved on behalf of the Council of Standards Australia on 22 September 1994 and on behalf of the Council of Standards New Zealand on 9 May 1994. It was published on 5 January 1995.

The following interests are represented on Committee IT/1:

Australian Bankers Association
 Australian Bureau of Statistics
 Australian Chamber of Commerce and Industry
 Australian Committee of Directors and Principals
 Australian Computer Society
 Australian Computer Users Association
 Australian Information Industry Association
 Australian Vice Chancellors Committee
 Department of Defence, Australia
 Department of Industry, Science and Technology, Australia
 Government Computing Service, New Zealand
 Information Exchange Steering Committee, Australia
 Institute of Information and Communication Technologies, CSIRO, Australia
 OPTUS Communications, Australia
 Telecom Australia
 Telecom New Zealand

Review of Standards. To keep abreast of progress in industry, Joint Australian/New Zealand Standards are subject to periodic review and are kept up to date by the issue of amendments or new editions as necessary. It is important therefore that Standards users ensure that they are in possession of the latest edition, and any amendments thereto.

Full details of all Joint Standards and related publications will be found in the Standards Australia and Standards New Zealand Catalogue of Publications; this information is supplemented each month by the magazines 'The Australian Standard' and 'Standards New Zealand', which subscribing members receive, and which give details of new publications, new editions and amendments, and of withdrawn Standards.

Suggestions for improvements to Joint Standards, addressed to the head office of either Standards Australia or Standards New Zealand, are welcomed. Notification of any inaccuracy or ambiguity found in a Joint Australian/New Zealand Standard should be made without delay in order that the matter may be investigated and appropriate action taken.

AS/NZS 3646.4 Supp 1:1995

**AS/NZS 3646.4
Supplement 1:1995**

**Information processing systems—
Open Systems Interconnection—
File transfer, access and
management**

**Part 4: File protocol specification—
Overlapped access**

**(Supplement to
AS/NZS 3646.4:1993)**

First published as Joint Standard
AS/NZS 3646.4 Supp 1:1995.

PUBLISHED JOINTLY BY:

STANDARDS AUSTRALIA
1 The Crescent,
Homebush NSW 2140 Australia

STANDARDS NEW ZEALAND
Level 10, Standards House,
155 The Terrace,
Wellington 6001 New Zealand

PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT/1 on Information Systems—Interconnection. It is identical with and has been reproduced from ISO 8571-4:1988/Amd.2:1993, *Information processing systems—Open Systems Interconnection—File Transfer, Access and Management, Part 4: File Protocol Specification, Amendment 2: Overlapped access*.

The objective of this Standard is to provide users of the FTAM Standards with an extension of the scope to include the services offered by overlapped access.

This Standard adds to the text of AS/NZS 3646.4:1993.

This Standard is one of a series of Open Systems Interconnection (OSI) Standards which are currently under development. Since OSI Standards are developmental, there may be some minor difficulties encountered in their implementation. For this reason, Standards Australia will be providing, through the OSI Help Desk, a service to coordinate and disseminate information concerning difficulties which are identified in using this Standard.

Users of this Standard are advised by Standards Australia and Standards New Zealand, under arrangements made with ISO and IEC, as well as certain other Standards organizations, that the number of this Standard is not reproduced on each page; its identity is shown only on the cover and title pages.

For the purpose of this Standard, the source text should be modified as follows:

- (a) *Terminology* The words 'this Australian/New Zealand Standard' should replace the words 'this International Standard' wherever they appear.
- (b) *Decimal marker* Substitute a full point for a comma where it appears as a decimal marker.

© Copyright – STANDARDS AUSTRALIA/STANDARDS NEW ZEALAND

Users of Standards are reminded that copyright subsists in all Standards Australia and Standards New Zealand publications and software. Except where the Copyright Act allows and except where provided for below no publications or software produced by Standards Australia or Standards New Zealand may be reproduced, stored in a retrieval system in any form or transmitted by any means without prior permission in writing from Standards Australia or Standards New Zealand. Permission may be conditional on an appropriate royalty payment. Australian requests for permission and information on commercial software royalties should be directed to the head office of Standards Australia. New Zealand requests should be directed to Standards New Zealand.

Up to 10 percent of the technical content pages of a Standard may be copied for use exclusively in-house by purchasers of the Standard without payment of a royalty or advice to Standards Australia or Standards New Zealand.

Inclusion of copyright material in computer software programs is also permitted without royalty payment provided such programs are used exclusively in-house by the creators of the programs.

Care should be taken to ensure that material used is from the current edition of the Standard and that it is updated whenever the Standard is amended or revised. The number and date of the Standard should therefore be clearly identified.

The use of material in print form or in computer software programs to be used commercially, with or without payment, or in commercial contracts is subject to the payment of a royalty. This policy may be varied by Standards Australia or Standards New Zealand at any time.

AUSTRALIAN/NEW ZEALAND STANDARD

**Information processing systems—Open Systems Interconnection—
File transfer, access and management****Part 4: File protocol specification—Overlapped access**

(Supplement to AS/NZS 3646.4:1993)

0. Introduction

Clause 0 provides an introduction to this amendment. The text in this clause is not intended for inclusion in ISO 8571 part 4.

0.1 General

ISO 8571 part 4 provides specifications of the protocols that support the internal and external file service interfaces.

This amendment extends these protocol specifications to provide support for the services offered by overlapped access.

0.2 Rationale

The objective in introducing overlapped access is to allow more efficient access to structured files when a single initiator has a need to perform many reading and updating operations; the serial nature of the current FTAM data transfer services introduces a significant control overhead if the FADUs are small. In this context, an FADU is small if its transmission time is comparable with the time to complete a confirmed service on the association (the association's round trip delay).

0.3 Summary

The current design envelope that there should be at most one file selection per association and one file open per file selection is maintained. If access to more than one file is to be overlapped, more than one association is necessary. The overlapped access takes place within a constant set of presentation contexts established as at present when the file is opened, or previously.

Two different degrees of overlap have been identified. Firstly, requests for future accesses may be issued whilst a previously requested BDT action is in progress, allowing the creation of a queue of read and write requests. In general, PCI relating to a given BDT action may be overlapped with

other BDT actions, subject to restrictions; this is called consecutive access. Secondly, read and write actions can be performed in parallel, so that both directions of data transfer are exploited at any one time. Requests are then taken from the queue whenever either direction of transfer becomes free. This is called concurrent access.

The transfer of a single FADU, specified in a single F-READ request has the same interpretation as in ISO 8571. The resultant effect on the virtual filestore of a set of overlapped requests using consecutive access shall be the same as that of the equivalent set of requests issued in series; the service provided is serializable. If concurrent access is used then the resultant effect of a set of write actions on the virtual filestore, is also serializable. However, due to the non-determinism introduced by the use of concurrent access, it is also possible that in some uses of the service, the data transferred as a result of a read action is not consistent with the current state of the file.

1. Scope

This amendment makes no additions to clause 1.

2. Field of application

This amendment makes no additions to clause 2.

3. References

This amendment makes no additions to clause 3.

4. Definitions and abbreviations

This amendment makes no additions to clause 4.

Section one: General

5. Overview of the protocol

5.2 Services assumed by the basic file protocol

Amend table 1.

5.6 Protocol functional units

Add to end of list.

- k) consecutive access functional unit
- l) concurrent access functional unit

FTAM Functional Unit	Session Functional Unit	Presentation Functional Unit
Kernel(4)	Kernel Duplex Optionally: Resynch(1) Minor synch(2)	Kernel Duplex Optionally: Resynch(1) Minor synch(2) Context Management(3)
Recovery	Minor Synch Symmetric Synchronize (5)	Minor Synch Symmetric Synchronize (5)
Restart	Minor Synch Resynchronize Symmetric Synchronize (5)	Minor Synch Resynchronize Symmetric Synchronize (5)

NOTES
5 The Symmetric Synchronize functional unit is used when overlapped access is in use. Pending the specification of presentation symmetric synchronisation services, recovery mechanisms will not be available for use during overlapped access.

Section two: Basic file protocol

6. State of the association provided

6.2 Additional state information

6.2.3 Next state indicator

Replace second sentence.

The defined values are the state names "initialised", "selected", "normal data transfer idle", "consecutive data transfer idle", "concurrent data transfer idle", and "unset".

6.2.5

Add as last paragraph:

In overlapped access, the bulk transfer number identifies the bulk data transfer that can be cancelled, restarted or recovered. Thus, the bulk transfer number is incremented only when there is an outstanding data transfer request and the previous data transfer cannot be cancelled, restarted or recovered. If an initiator and a responder have different bulk transfer numbers then it is the bulk data transfer associated with the smaller of the two numbers that is cancelled, restarted or recovered.

6.2.6 Transfer number

In overlapped access, the transfer number identifies the bulk data transfer within a sequence of transfers from one data transfer idle state to a next data transfer idle state within an open regime. It is set to zero at each data transfer idle state.

In concurrent overlapped access, two transfer numbers are maintained - one for reads and one for writes.

7. File protocol data units

This amendment makes no additions to clause 7.

8. File initiating entity actions

9. File responding entity actions

10. File general actions

This amendment makes no additions to clause 10.

Section three: The basic bulk data transfer protocol

11. State of bulk data transfer activity

Re-label sub-clause 11.2.

11.2 Additional state of the entities (without overlapped access)

Replace first paragraph.

The following sub-clauses define the items of state information associated with the basic protocol entities for the purposes of bulk data transfer without overlapped access.

Add the following sub-clauses.

11.3 Additional state of the entities (with consecutive access)

The following sub-clauses define the items of state information associated with the basic protocol entities for the purposes of bulk data transfer with consecutive access.

11.3.1 Current Transfer Number

The current transfer number indicates the transfer number of the bulk data transfer procedure that is currently considered "in progress".

11.3.2 Start Bulk Transfer Number

The start bulk transfer number indicates the bulk transfer number of the first in a sequence of overlapped data transfers. The start bulk transfer number is used to calculate the bulk transfer number and transfer number for cancel, recover and restart.

11.3.3 Checkpoint identifier expected

The checkpoint identifier expected reflects the sequence of checkpoints within bulk data, and is reset by the start of the

bulk data transfer, and by error recovery mechanisms. The value is incremented when a checkpoint is made. The value is an integer in the range 1 to 999998. Initially, the value is determined by the state of the association.

The checkpoint identifier expected applies only to FTAM regimes for which the use of the restart and/or recovery functional units have been successfully negotiated.

11.3.4 First next synchronisation point number

The first next synchronisation point number reflects the sequence of events in the supporting synchronisation services for a sender. The number is the serial number of the next session synchronisation point to be issued by the session service provider. The value is an integer in the range 0 to 999998. Initially on a newly created session connection, the value is 1.

The first next synchronisation point number applies only to FTAM regimes for which the use of the presentation symmetric synchronisation functional unit has been successfully negotiated.

11.3.5 Second next synchronisation point number

The second next synchronisation point number reflects the sequence of events in the supporting synchronisation services for a receiver. The number is the serial number of the next session synchronisation point to be issued by the session service provider. The value is an integer in the range 0 to 999998. Initially on a newly created session connection, the value is 1.

The second next synchronisation point number applies only to FTAM regimes for which the use of the presentation symmetric synchronisation functional unit has been successfully negotiated.

Table 5 - Protocol Data Units

Name	Carried by	Functional units
F-CHECK request (see note 2)	P-SYNC-MINOR request	recovery, restart
F-CHECK response (see note 2)	P-SYNC-MINOR response	recovery, restart

NOTES

1 The data value corresponds to an F-DATA request service primitive. There is no F-DATA request PDU as such.

2 If overlapped access is not in use then the F-CHECK request and response primitives are mapped directly onto the P-SYNC-MINOR request and response primitives, with no additional syntax.

11.3.6 Synchronisation offset

The synchronisation offset is a constant established when a read or write bulk data transfer is initiated or recovered, which gives the difference between the checkpoint identifier expected and the next resynchronisation point number.

The synchronisation offset applies only to FTAM regimes for which use of the presentation symmetric synchronisation functional unit has been successfully negotiated.

NOTE - Except during the issue of a checkpoint, or performance of the restart procedure the value of the synchronisation offset is equal to the difference between the expected checkpoint number and the next synchronisation point number.

11.3.7 Outstanding Checkpoint counter

The outstanding checkpoint counter records the number of checkpoints which are unacknowledged.

11.3.8 Read/Write Indicator

The read/write indicator records whether the current bulk data transfer is to or from the initiator. The value is set upon the beginning of the data transfer regime. The defined values are "reading", "writing" and "unset". The initial value is unset.

11.3.9 Discard Indicator

The discard indicator is used to signal that data received during cancellation or before recovery is invalid and should be thrown away. If the recovery or restart functional units are selected and presentation resynchronisation functional unit has been successfully negotiated, it is used in conjunction with session resynchronisation to produce a recovery without user visibility of the error; otherwise it is used during the cancel phase. The defined values are "unset" and "set". Initially, the value is "unset".

11.3.10 Transfer Request Queue

The transfer request queue records the transfer number and transfer type (read or write) of all outstanding data transfers.

11.3.11 Transfer End Queue

The transfer end queue records the transfer numbers of all data transfers, already recorded on the transfer request queue, for which a transfer end PDU request or response has been issued.

11.3.12 Data End Queue

The data end queue records the transfer numbers of all data transfers, already recorded on the data request queue, for which a data end PDU has been issued or received.

11.3.13 Read and write checkpoint tables

The read and write checkpoint tables (two separate tables are kept) record checkpointing information for each ongoing bulk data transfer. The following information is kept for each currently active bulk data transfer: transfer number, bulk transfer number, checkpoint expected, synchronisation offset, and checkpoint counter. The size of the table is that of the transfer window negotiated when the file is opened.

11.3.14 Last transfer end confirm indicator

The last transfer end confirm indicator records the last transfer end received by the initiator. The last transfer end confirm received is sent to the responder on a transfer end request PDU. It is also included on cancel, recover and restart PDUs. The responder uses the information in removing items from the transfer end response queue and in the re-issuing of transfer-end response after a session resynchronisation in the direction of responder to initiator.

11.3.15 Last transfer end indication indicator

The last transfer end indication indicator records the last transfer end indication received by the responder. It is included on cancel, recover and restart PDUs. The initiator uses the information in the re-issuing of transfer end requests after a session resynchronisation in the direction of initiator to responder.

11.4 Additional state of the entities (with concurrent access)

The following sub-clauses define the items of state information associated with the basic protocol entities for the purposes of bulk data transfer with concurrent access.

11.4.1 Current Read Transfer Number and Current Write Transfer Number

The current transfer read number and the current transfer write number indicate the transfer numbers of the read and write data transfer procedures that are in progress.

11.4.2 Start Bulk Transfer Number

The start bulk transfer number indicates the bulk transfer number of the first in a sequence of overlapped data transfers. The start bulk transfer number is used to calculate the bulk transfer number and transfer number for cancel, recover, and restart.

11.4.3 Checkpoint identifier expected

The checkpoint identifier expected reflects the sequence of checkpoints within bulk data, and is reset by the start of the bulk data transfer, and by error recovery mechanisms. The value is incremented when a checkpoint is made. The value

is an integer in the range 1 to 999998. Initially, the value is determined by the state of the association.

The checkpoint identifier expected applies only to FTAM regimes for which the use of the restart and/or recovery functional units have been successfully negotiated.

11.4.4 First next synchronisation point number

The first next synchronisation point number reflects the sequence of events in the supporting synchronisation services for a sender. The number is the serial number of the next session synchronisation point to be issued by the session service provider. The value is an integer in the range 0 to 999998. Initially on a newly created session connection, the value is 1.

The first next synchronisation point number applies only to FTAM regimes for which the use of the presentation symmetric synchronisation functional unit has been successfully negotiated.

11.4.5 Second next synchronisation point number

The second next synchronisation point number reflects the sequence of events in the supporting synchronisation services for a receiver. The number is the serial number of the next session synchronisation point to be issued by the session service provider. The value is an integer in the range 0 to 999998. Initially on a newly created session connection, the value is 1.

The second next synchronisation point number applies only to FTAM regimes for which the use of the presentation symmetric synchronisation functional unit has been successfully negotiated.

11.4.6 Synchronisation offset

The synchronisation offset is a constant established when a read or write bulk data transfer is initiated or recovered, which gives the difference between the checkpoint identifier expected and the next resynchronisation point number.

The synchronisation offset applies only to FTAM regimes for which use of the presentation symmetric synchronisation functional unit has been successfully negotiated.

NOTE - Except during the issue of a checkpoint, or performance of the restart procedure the value of the synchronisation offset is equal to the difference between the expected checkpoint number and the next synchronisation point number.

11.4.7 Outstanding Checkpoint counter

The outstanding checkpoint counter records the number of

checkpoints which are unacknowledged.

11.4.8 Read Indicator and Write Indicator

The read indicator and the write indicator record whether or not there are read or write data transfers in progress. The defined values are "reading"/"writing" and "unset". The initial value is unset.

11.4.9 Discard indicator

The discard indicator is used to signal that data received during cancellation or before recovery is invalid and should be thrown away. If the recovery or restart functional units are selected and presentation resynchronisation functional unit has been successfully negotiated, it is used in conjunction with session resynchronisation to produce a recovery without user visibility of the error; otherwise it is used during the cancel phase. The defined values are "unset" and "set". Initially, the value is "unset".

11.4.10 Transfer Read Request Queue and Transfer Write Request Queue

The transfer read request queue and the transfer write request queue record the transfer numbers of all outstanding read and write data transfers.

11.4.11 Transfer End Read Queue and Transfer End Write Queue

The transfer end read queue and the transfer end write queue record the transfer numbers of all read and write data transfers, already recorded on the transfer request read (write) queue, for which a transfer end PDU request or response has been issued.

11.4.12 Data End Read Queue and Data End Write Queue

The data end read queue and the data end write queue record the transfer numbers of all read and write data transfers, all ready recorded on the transfer request read (write) queue, for which a data end PDU has been issued/received.

11.4.13 Read and write checkpoint tables

The read and write checkpoint tables (two separate tables are kept) record checkpointing information for each ongoing bulk data transfer. The following information is kept for each currently active bulk data transfer: transfer number, bulk transfer number, checkpoint expected, synchronisation offset, and checkpoint counter. The size of the table is that of the transfer window negotiated when the file is opened.

11.4.14 Last transfer end confirm indicator

The last transfer end confirm indicator records the last

transfer end read (write) received by the initiator. The last transfer end confirm received is sent to the responder on a transfer end request PDU. It is also included on cancel, recover and restart PDUs. The responder uses the information in removing items from the transfer end response queue and in the re-issuing of transfer-end response after a session resynchronisation in the direction of responder to initiator.

11.4.15 Last transfer end indication indicator

The last transfer end indication indicator records the last transfer end read (write) indication received by the responder. It is included on cancel, recover and restart PDUs. The initiator uses the information in the re-issuing of transfer end requests after a session resynchronisation in the direction of initiator to responder.

12. Bulk data transfer protocol data units

Amend table 5.

13. Bulk data transfer initiating entity actions

14. Bulk data transfer responding entity actions

15. Bulk data transfer sending entity actions

16. Bulk data transfer receiving entity actions

17. Bulk data transfer general actions

Section four: The error recovery protocol

18. Protocol mechanisms

19. Specification of the error control protocol

Section five: Abstract Syntax

20. Abstract Syntax Definition

20.3 ASN.1 Module Definition

Amend Figure 7.

Amend Figure 8.

Amend Figure 9.

Amend Figure 10.

21. Application Context Name

This amendment makes no additions to clause 21.

22. Conformance

This amendment makes no additions to clause 22.

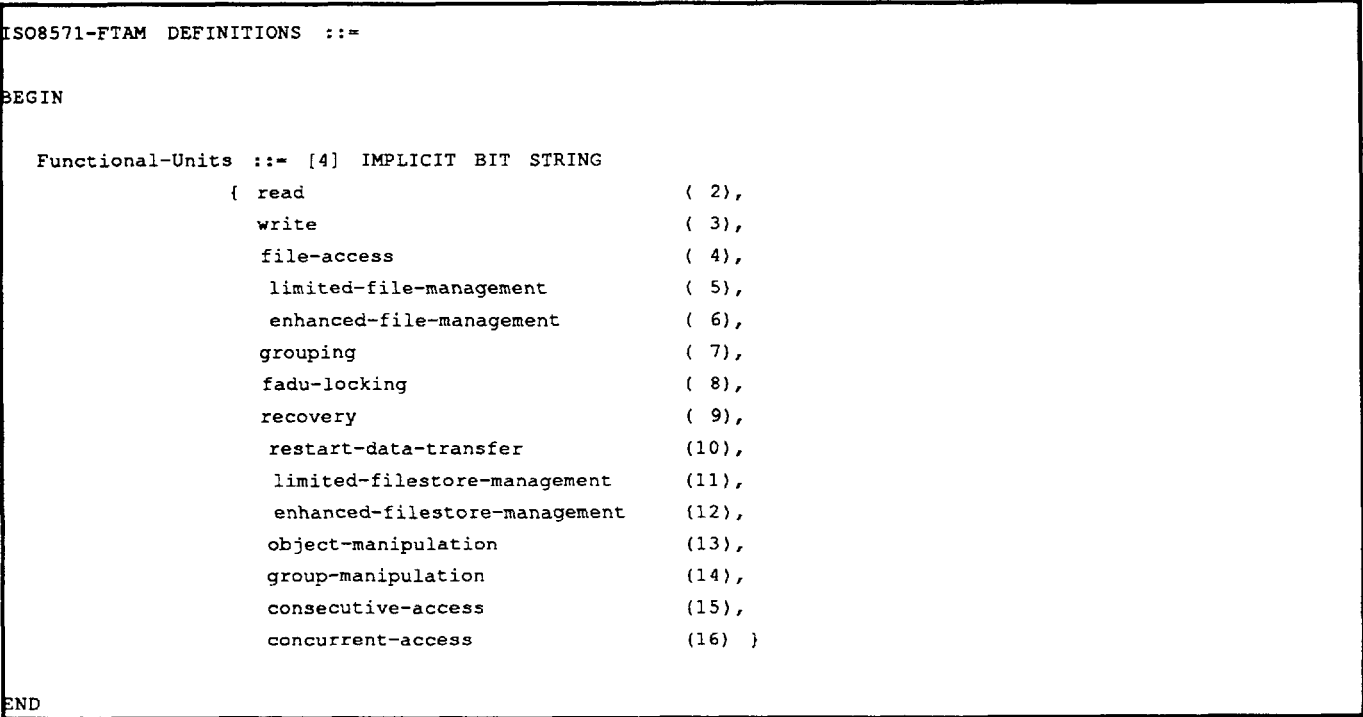


Figure 7 - FTAM regime PDUs

ISO8571-FTAM DEFINITIONS :-

BEGIN

```

F-OPEN-request ::= SEQUENCE {
    processing-mode          [0] IMPLICIT BIT STRING
                               { f-read          (0),
                                 f-insert        (1),
                                 f-replace       (2),
                                 f-extend        (3),
                                 f-erase         (4) } DEFAULT { f-read },
    contents-type            [1] CHOICE {
        unknown              [0] IMPLICIT NULL,
        proposed              [1] Contents-Type-Attribute },
    concurrency-control       Concurrency-Control OPTIONAL,
    shared-ASE-information    Shared-ASE-Information OPTIONAL,
    enable-fadu-locking       [2] IMPLICIT BOOLEAN DEFAULT FALSE,
    activity-identifier       Activity-Identifier OPTIONAL,
        - Only used in the recovery functional unit.
    recovery-mode             [3] IMPLICIT INTEGER
                               { none (0),
                                 at-start-of-file (1),
                                 at-any-active-checkpoint (2) } DEFAULT none,
    remove-contexts           [4] IMPLICIT SET OF Abstract-Syntax-Name OPTIONAL,
    define-contexts           [5] IMPLICIT SET OF Abstract-Syntax-Name OPTIONAL,
        - The following are conditional on the negotiation of the consecutive overlap or
        - concurrent overlap functional units.
    degree-of-overlap         Degree-Of-Overlap OPTIONAL,
    transfer-window           [7] IMPLICIT INTEGER OPTIONAL }

F-OPEN-response ::= SEQUENCE {
    state-result              State-Result DEFAULT success,
    action-result             Action-Result DEFAULT success,
    contents-type             [1] Contents-Type-Attribute,
    concurrency-control        Concurrency-Control OPTIONAL,
    shared-ASE-information     Shared-ASE-Information OPTIONAL,
    diagnostic                 Diagnostic OPTIONAL,
    recovery-mode              [3] IMPLICIT INTEGER
                               { none (0),
                                 at-start-of-file (1),
                                 at-any-active-checkpoint (2) } DEFAULT none,
    presentation-action        [6] IMPLICIT BOOLEAN DEFAULT FALSE,
        - This flag is set if the responder is going to follow this response by a P-ALTER-CONTEXT
        - exchange.
        - The following are conditional on the negotiation of the consecutive access or
        - concurrent access functional units.
    degree-of-overlap         Degree-Of-Overlap OPTIONAL,
    transfer-window           [7] IMPLICIT INTEGER OPTIONAL }

```

Figure 8 - File selection and file open regime PDUs

```

F-RECOVER-request ::= SEQUENCE {
    activity-identifier          Activity-Identifier,
    bulk-transfer-number         [0] IMPLICIT INTEGER,
        - If concurrent access was in use then this parameter indicates the read bulk
        - transfer.
    requested-access             Access-Request,
    access-passwords             Access-Passwords OPTIONAL,
    recovery-point               [2] IMPLICIT INTEGER DEFAULT 0,
        - zero indicates beginning of file point after last checkpoint indicates end of file
    remove-contexts              [3] IMPLICIT SET OF Abstract-Syntax-Name OPTIONAL,
    define-contexts              [4] IMPLICIT SET OF Abstract-Syntax-Name OPTIONAL,
        - The following are conditional on the negotiation of overlapped access.
    concurrent-bulk-transfer-number [7] IMPLICIT INTEGER OPTIONAL,
        - conditional on use of concurrent access
    concurrent-recovery-point     [8] IMPLICIT INTEGER OPTIONAL,
        - conditional on use of concurrent access. Zero indicates beginning of file
        - point after last checkpoint indicates end of file
    last-transfer-end-read-response [9] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-response [10] IMPLICIT INTEGER OPTIONAL }

F-RECOVER-response ::= SEQUENCE {
    state-result                 State-Result Default success,
    action-result                Action-Result DEFAULT success,
    contents-type                [1] Contents-Type-Attribute,
    recovery-point               [2] IMPLICIT INTEGER DEFAULT 0,
        - Zero indicates beginning of file; point after last checkpoint indicates end of file
    diagnostic                   Diagnostic OPTIONAL,
    presentation-action          [6] IMPLICIT BOOLEAN DEFAULT FALSE,
        - This flag is set if the responder is going to follow this response
        - by a P-ALTER-CONTEXT exchange.
        - The following are conditional on the negotiation of overlapped access.
    concurrent-recovery-point     [8] IMPLICIT INTEGER OPTIONAL,
        - conditional on use of concurrent access. Zero indicates beginning of file; point after
        - last checkpoint indicates end of file
    last-transfer-end-read-request [9] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-request [10] IMPLICIT INTEGER OPTIONAL }

```

END

Figure 8 (continued) - File selection and file open regime PDUs

Amend Figure 9.

```

ISO8571-FTAM DEFINITIONS ::

BEGIN

F-READ-request ::= SEQUENCE {
    file-access-data-unit-identity FADU-Identity,
    access-context                  Access-Context,
    fadu-lock                       FADU-Lock OPTIONAL,
    - The following is conditional on the negotiation of consecutive of concurrent access.
    transfer-number                 [0] IMPLICIT INTEGER OPTIONAL }

F-WRITE-request ::= SEQUENCE {
    file-access-data-unit-operation [0] IMPLICIT INTEGER
        { insert    (0),
          replace   (1),
          extend    (2) },
    file-access-data-unit-identity FADU-Identity,
    fadu-lock                       FADU-Lock OPTIONAL,
    - The following is conditional on the negotiation of consecutive or concurrent access.
    transfer-number                 [1] IMPLICIT INTEGER OPTIONAL }

F-TRANSFER-END-request ::= SEQUENCE {
    shared-ASE-information          Shared-ASE-Information OPTIONAL,
    - The following are conditional on the negotiation of consecutive or concurrent access.
    request-type                   Request-Type OPTIONAL,
    transfer-number                 [0] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-read-response [1] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-response [2] IMPLICIT INTEGER OPTIONAL }

F-TRANSFER-END-response ::= SEQUENCE {
    action-result                  Action-Result DEFAULT success,
    shared-ASE-information         Shared-ASE-Information OPTIONAL,
    diagnostic                     Diagnostic OPTIONAL,
    - The following are conditional on the negotiation of consecutive or concurrent access.
    request-type                   Request-Type OPTIONAL,
    transfer-number                 [0] IMPLICIT INTEGER OPTIONAL }

F-CANCEL-request ::= SEQUENCE {
    action-result                  Action-Result DEFAULT success,
    shared-ASE-information         Shared-ASE-Information OPTIONAL,
    diagnostic                     Diagnostic OPTIONAL,
    - The following are conditional on the negotiation of consecutive or concurrent access.
    request-type                   Request-Type,
    transfer-number                 [0] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-read-request [1] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-read-response [2] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-request [3] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-response [4] IMPLICIT INTEGER OPTIONAL }

F-CANCEL-response ::= SEQUENCE {
    action-result                  Action-Result DEFAULT success,
    shared-ASE-information         Shared-ASE-Information OPTIONAL,

```

Figure 9 - Bulk data transfer PDUs

```

diagnostic                                Diagnostic OPTIONAL,
    - The following are conditional on the negotiation of consecutive or concurrent access.
request-type                             Request-Type OPTIONAL,
transfer-number                           [0] IMPLICIT INTEGER OPTIONAL,
last-transfer-end-read-request [1] IMPLICIT INTEGER OPTIONAL,
last-transfer-end-read-response [2] IMPLICIT INTEGER OPTIONAL,
last-transfer-end-write-request [3] IMPLICIT INTEGER OPTIONAL,
last-transfer-end-write-response [4] IMPLICIT INTEGER OPTIONAL }

F-CHECK-request ::= SEQUENCE {
    checkpoint-identifier [0] IMPLICIT INTEGER,
    transfer-number [1] IMPLICIT INTEGER }

F-CHECK-response ::= SEQUENCE {
    checkpoint-identifier [0] IMPLICIT INTEGER,
    transfer-number [1] IMPLICIT INTEGER }

F-RESTART-request ::= SEQUENCE {
    checkpoint-identifier [0] IMPLICIT INTEGER,
    - The following are conditional on the negotiation of consecutive or concurrent access.
    request-type Request-Type OPTIONAL,
    transfer-number [1] IMPLICIT INTEGER,
    last-transfer-end-read-request [2] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-read-response [3] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-request [4] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-response [5] IMPLICIT INTEGER OPTIONAL }

F-RESTART-response ::= SEQUENCE {
    checkpoint-identifier [0] IMPLICIT INTEGER,
    - The following are conditional on the negotiation of consecutive or concurrent access.
    request-type Request-Type OPTIONAL,
    transfer-number [1] IMPLICIT INTEGER,
    last-transfer-end-read-request [2] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-read-response [3] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-request [4] IMPLICIT INTEGER OPTIONAL,
    last-transfer-end-write-response [5] IMPLICIT INTEGER OPTIONAL }
END

```

Figure 9 (continued) - Bulk data transfer PDUs

```
ISO8571-FTAM DEFINITIONS ::=
BEGIN

  Degree-Of-Overlap ::= [Application 30] IMPLICIT INTEGER
    { normal      (0),
      consecutive (1),
      concurrent  (2) }

  Request-Type ::= [Application 31] IMPLICIT INTEGER
    { read (0),
      write (1) }

END
```

Figure 10 - Application wide types

Section six: Conformance

Annex A

Protocol state tables

(This annex forms part of the standard.)

A.1 Introduction

"W" Write

Add note after third paragraph:

Note - for concurrent access, there are separate state tables for read and write procedures, for both the FERPM and the Basic FPM.

A.1.1 System model

Add last paragraph.

In addition, for concurrent overlap, both the FPM and the FERPM are considered to consist of two sub-entities of read FPM and write FPM, and read FERPM and write FERPM. The additional local signals for concurrent overlap are:

- q) L-TRERQ-R - signal indicating a re-issued F-TRANSFER-END(read) request primitive;
- r) L-TRERQ-W -signal indicating a re-issued F-TRANSFER-END(write) request primitive;
- s) L-TRERP-W - signal indicating a re-issued F-TRANSFER-END(write) response primitive;
- t) L-TRERP-R - signal indicating a re-issued F-TRANSFER-END(read) response primitive;
- u) L-WRTRQ - signal indicating a re-issued F-WRITE request primitive;
- v) L-REARQ - signal indicating a re-issued F-READ request primitive;
- w) L-OPEN - signal to a sub-entity of successful processing of F-OPEN primitive;
- x) L-CLOSE - signal to a sub-entity of successful processing of F-CLOSE primitive;
- y) L-RECVR - signal to a sub-entity to go into or out of the RECOVER-WT state;
- z) L-PASIVE - signal to a sub-entity to go into the PASSIVE state.

A.1.2 Incoming states

Add last paragraph.

- h) for concurrent overlap, the following suffices indicate the data transfer types of primitives and PDUs:

"R" Read

A.1.4 States

Add last paragraph.

The prefix "Q-" indicates the use of consecutive access. The prefixes "K-R-" & "K-W-" indicate the use of concurrent access and are with respect to read and writes respectively.

A.1.5 Predicates

Add to functional units of FPM.

- U15: consecutive overlap
- U16: concurrent overlap

A.1.8 Additional State Information

Replace first sentence.

The tables make use of the indicators and other state variables defined in: 6.2, 11.2 and 11.3.

Add last paragraph.

In overlapped access the following symbols are used:

- TN - transfer number
- RTN - read transfer number
- WTN - write transfer number
- BTN - bulk transfer number
- SBTN - start bulk transfer number
- RSTTN - restart transfer number
- TE - transfer end

Add the following sub-clauses:

A.1.8.1 FPM in overlapped access

Each transfer request is placed in a queue. It is removed from the queue when a F-TRANSFER-END response is issued or a F-TRANSFER-END confirm is received.

The current state of the FPM always represents the state of the transfer that is currently on the top of the request queue. The FPM places each F-DATA-END primitive that is issued or received on a queue and checks the next request in the request queue to set the read/write indicator(s) for the next transfer.

During the exchange of F-TRANSFER-END primitives, the initiating FPM places the F-TRANSFER-END request in a queue and the responding FPM places the F-TRANSFER-END response in a queue (these are saved for possible re-issuing when a re-synchronisation occurs). When a F-TRANSFER-END confirm is received then the F-TRANSFER-END request is removed from its queue. A F-TRANSFER-END response is removed from its queue when the transfer end response parameter on a F-TRANSFER-END request indicates its receipt by the initiator.

The transfer-end queues, the data-end queues and the read/write indicator(s) are used to determine the next state of the FPM.

To make the state tables manageable for concurrent overlap the basic FPM is considered to consist of the two sub-entities of read FPM and write FPM. Each sub-entity maintains its own states and communicates with the other via global variables. The operation of the sub-entities are independent apart from during cancel and restart. The cancellation or restart of a write may destroy control information of data transfer and transfer end requests for reads; the cancellation or restart of a read may destroy control information of transfer end responses for writes. These have to be saved and re-issued at the completion of resynchronisation to maintain the independence of read and write activities.

A.1.8.2 FERPM in overlapped access

The FERPM is divided into two sub-entities of read and write in concurrent access. It is assumed that under normal operation the read sub-entity will look after the FTAM regime and the file management regime. The write sub-entity stays in the PASSIVE state and enters into the XFER-IDLE state when it receives the local signal L-OPEN (from the read sub-entity) and it will go back to the PASSIVE state when it receives the locals signal L-CLOSE (read sub-entity). Either sub-entity can perform recovery of the FTAM regime and the file management regime. The sub-entity that is performing the recovery will send the local signal L-RECV, to put the other entity into the RECOVER-WT state. When recovery is completed, the recovering sub-entity will again issue a L-RECV signal to inform the other sub-entity that recovery is completed. It is assumed that each sub-entity will be informed by different local signals for type 1 and type 2 errors. The same signal will be sent to both sub-entities when type 3 errors occur. The re-sending of any control information is done at the end of the restart phase.

When a write sub-entity receives a type 2 error it will issue a Cancel in write and a read sub-entity will issue a Cancel in read when it receives a type 2 error. Recovery actions do not start until both cancels are completed. The entity that completes the cancel first will be put into the RECOVER-WT state. An indicator is set to remember which entity is in the wait state.

The FERPM is given the task of re-issuing F-TRANSFER-END requests and responses and F-READ and F-WRITE

requests after a session resynchronisation. The cancellation of a write will require the initiator to re-issue any possible read requests and transfer end (read) requests. The cancellation of a read will require the responder to re-issue any possible transfer end (write) responses. In addition, during error recovery the initiator will resend any write requests from the transfer request queue and transfer end (write) requests from its docket as well as any from its transfer end queue. The responder will resend and transfer end (read) response from its docket and from its transfer end queue. It is assumed that F-TRANSFER-END primitives will be re-issued before transfer request primitives.

To reduce the complexity of the state tables, all processing of user primitives are held until the completion of error recovery.

A.1.9 Additional processing indicators for the FERPM

During recovery from class II and III errors in concurrent access, the FERPM uses the following extra indicators to co-ordinate events between the two sub-entities.

- a) Error3 indicator - it is set when a sub-entity has sent a local L-ERROR3 signal to the other sub-entity to indicate that it is unable to recover from a class II error and is treating the error as a class III error. It is unset when recovery is completed.
- b) Write Cancel indicator - it is set when the write sub-entity has issued a CANCEL when an error II occurs. It is unset when recovery is completed.
- c) Read Cancel indicator - it is set when the read sub-entity has issued a CANCEL when an error II occurs. It is unset when recovery is completed.
- d) Recovery indicator - It is set to READ if a read sub-entity is passive during the recovery from class IV/III errors. It is set to WRITE if a write sub-entity is passive during the recovery from class IV/III errors.

A.2 FTAM regime management protocol machine (Kernel functional unit)

...

A.3 File regime management protocol machine

...

A.4 Bulk data transfer protocol machine

A.4.1 States - bulk data transfer

Relabel A.4.1:

A.4.1.1 Normal access (non-overlapped)

Add the following sub-clauses.

A.4.1.2 Consecutive access

Q-DXFRIDLE	Data transfer idle in consecutive overlap
Q-READ	Read data transfer in consecutive overlap
Q-I-READ-ENDING	Read transfer end, wait for F-TRANSFER-END request primitive from the internal file service user in consecutive overlap
Q-READ-ENDING	Read transfer ending, wait for transfer end request PDU in consecutive overlap
Q-I-R-XFER-ENDING	Wait for F-TRANSFER-END response primitive after READ from the internal file service user in consecutive overlap
Q-R-XFER-ENDING	Wait for TRANSFER-END response PDU after READ in consecutive overlap
Q-WRITE	Write data transfer in consecutive overlap
Q-I-WRT-ENDING	Write TRANSFER-END, wait for TRANSFER-END request PDU in consecutive overlap
Q-WRT-ENDING	Write transfer-end, wait for transfer end request PDU.
Q-I-W-XFER-ENDING	Wait for F-TRANSFER-END response primitive after WRITE from the internal file service user in consecutive overlap
Q-W-XFER-ENDING	Wait for TRANSFER-END response PDU after write in consecutive overlap
Q-CANCEL-PD	Cancel pending, wait for cancel response PDU in consecutive overlap
Q-I-CANCEL-PD	Cancel pending, wait for F-CANCEL response primitive from the internal file service user in consecutive overlap
Q-RRESTART-PD	Read restart pending, wait for restart response PDU in read operation in consecutive overlap
Q-I-RRESTART-PD	Read restart pending, wait for F-RESTART response primitive in read operation from the internal file service user in consecutive overlap

Q-WRESTART-PD	Write restart pending, wait for restart response PDU in write operation in consecutive overlap
Q-I-WRESTART-PD	Write restart pending, wait for F-RESTART response primitive in write operation from the internal file service user in consecutive overlap
Q-XFRIDLE-REC	Data transfer file during recovery in consecutive overlap
Q-REA-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, restart requested in consecutive overlap
Q-RES-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, restart requested in consecutive overlap
Q-CAN-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, cancel requested in consecutive overlap
Q-WRT-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive after WRITE in consecutive overlap
Q-REA-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive after READ in consecutive overlap
Q-RES-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive, restart requested in consecutive overlap
Q-CAN-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive, cancel requested in consecutive overlap
Q-WRT-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive after WRITE in consecutive overlap
Q-RESTART-CAN-PD	Restart cancel pending, wait for cancel or restart response PDU in consecutive overlap

A.4.1.3 Concurrent access

States for READ entity:

K-R-DXFRIDLE	Data transfer idle in concurrent overlap in READ
K-READ	Read data transfer in concurrent overlap in READ
K-I-READ-ENDING	Read transfer end, wait for F-TRANSFER-END request primitive from the internal file service user in concurrent overlap in READ
K-READ-ENDING	Read transfer ending, wait for F-TRANSFER-END request PDU in concurrent overlap in READ
K-I-RXFER-ENDING	Wait for F-TRANSFER-END response primitive after READ from the internal file service user in concurrent overlap in READ
K-R-XFER-ENDING	Wait for F-TRANSFER-END response PDU after READ in concurrent overlap in READ
K-RCANCEL-PD	Cancel pending, wait for cancel response PDU in concurrent overlap in READ
K-I-RCANCEL-PD	Cancel pending, wait for F-CANCEL response primitive from the internal file service user in concurrent overlap in READ
K-RRESTART-PD	Read restart pending, wait for restart response PDU in read operation in concurrent overlap in READ
K-I-RRESTART-PD	Read restart pending, wait for F-RESTART response primitive in read operation from the internal file service user in concurrent overlap in READ
K-R-XFRIDLE-REC	Data transfer file during recover in concurrent overlap in READ
K-REA-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, restart requested in concurrent overlap in READ
K-RRES-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, restart requested in concurrent overlap in READ
K-RCAN-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, cancel requested in concurrent overlap in READ
K-RRES-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive, restart requested in concurrent overlap in READ
K-RCAN-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive, cancel requested in concurrent overlap in READ
K-REA-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive after READ in concurrent overlap in READ
K-RRESTART-CAN-PD	Restart cancel pending, wait for cancel or restart response PdU in concurrent overlap in READ

States for WRITE entity:

K-W-DXFRIDLE	Data transfer idle in concurrent overlap in WRITE
K-WRITE	Write data transfer in concurrent overlap in WRITE
K-I-WRT-ENDING	Write transfer end, wait for F-TRANSFER-END request PDU in concurrent overlap in WRITE

K-WRT-ENDING	Write transfer-end, wait for F-TRANSFER-ENDING request PDU in concurrent overlap in WRITE
K-I-WXFER-ENDING	Wait for F-TRANSFER-END response primitive after WRITE from the internal file service user in concurrent overlap in WRITE
K-W-XFER-ENDING	Wait for F-TRANSFER-END response PDU after WRITE in concurrent overlap in WRITE
K-WCANCEL-PD	Cancel pending, wait for cancel response PDU in concurrent overlap in WRITE
K-I-WCANCEL-PD	Cancel pending, wait for F-CANCEL response primitive from the internal file service user in concurrent overlap in WRITE
K-WRESTART-PD	Write restart pending, wait for restart response PDU in write operation in concurrent overlap in WRITE
K-I-WRESTART-PD	Write restart pending, wait for F-RESTART response primitive in write operation from the internal file service user in concurrent overlap in WRITE
K-W-XFRIDLE-REC	Data transfer file during recover in concurrent overlap in WRITE
K-WRT-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, restart requested in concurrent overlap in WRITE
K-WRES-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, restart requested in concurrent overlap in WRITE
K-WCAN-SYMIN-PD	Wait for first P-SYNC-MINOR indication primitive, cancel requested in concurrent overlap in WRITE
K-WRES-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive, restart requested in concurrent overlap in WRITE
K-WCAN-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive, cancel requested in concurrent overlap in WRITE
K-WRT-SYCMCF-PD	Wait for first P-SYNC-MINOR confirm primitive after WRITE in concurrent overlap in WRITE
K-WRESTART-CAN-PD	Restart cancel pending, wait for cancel or restart response PDU in concurrent overlap in WRITE

A.4.4 Specific actions - bulk data transfer

Relabel A.4.4:

A.4.4.1 Normal access (non-overlapped)

Add the following sub-clause.

A.4.4.2 Consecutive and Concurrent access

- [2] Add a PDU to the current PSDU and terminate the current PSDU.
- [15] Preserve the PDU for further processing
- [16] Add a PDU to the current PSDU and optionally terminate the current PSDU, according to local decision.
- [18] Add the data given on the F-DATA request to the current PSDU. Optionally (depending upon local system considerations) terminate the PSDU.
- [21] Unset discard indicator.
- [25] Terminate the current PSDU.
- [44] Increment the bulk transfer number.
- [47] Preserve the primitive parameters for subsequent processing. In some states, it happens that when F-CANCEL primitive is preserved, there is already an F-RESTART primitive preserved: in this case, the F-RESTART primitive shall be discarded.
- [401] Place request in request queue if request not already in the queue.
- [402] Add a PDU to the current PSDU and terminate the current PSDU or optionally send the PDU when the current PSDU is terminated depending upon local system considerations.
- [403] Use as parameter the user data from P-SYMIN.
- [404] Place transfer end request in TE request queue if request not already in the queue.
- [405] Remove first TE request from TE request queue and update the last TE confirm received and remove corresponding

entry for TN received in checkpoint table.

- [406] Remove first transfer request from transfer request queue.
- [407] Place current TN in DE request queue.
- [408] Remove first DE request from DE request queue.
- [409] Set indicator to read.
- [410] Set indicator to write.
- [411] Mark the next transfer request (to show that the current TN has been updated in the checkpoint table)
- [412] Add 1 to current transfer number and in checkpoint table corresponding to TN, set TN, BTN to be the current TN and BTN and set checkpoint counter to zero.
- [413] Set BTN to be SBTN.
- [414] Place current TN in DE indication queue.
- [415] Remove first DE indication from DE indication queue.
- [416] Send PDU with parameters: TN set to BTN - SBTN and TN of last TE confirm (or confirms) received as user data on a P-Resync request (or response) with second item set as abandon. (*init read)
- [417] Send PDU with parameters: TN set to BTN - SBTN and TN of last TE request (or requests) issued as user data on a P-Resync request (or response) with first item set as abandon. (*init write)
- [418] Send PDU with parameters: TN set to BTN - SBTN and TN of last TE response (or response) issued as user data on a P-Resync request (or response) with first item set as abandon. (*resp read)
- [419] Send PDU with parameters: TN set to BTN - SBTN and TN of last TE indication (or indications) received as user data on a P-Resync request (or response) with second item set as abandon. (*resp write)
- [420] Clear queues and variables except BTN and the NSPNs.
- [421] Reissue TE read requests and remove any TE write requests from TE request queue (or queues).
- [425] PDU with TN and the checkpoint identifier is sent as user data on a P-RESYNC response primitive with first resync item set to restart and the first resync point serial number equal to those received on the P-RESYNC indication
- [426] PDU with TN and the checkpoint identifier is sent as user data on a P-RESYNC request primitive with second resync item set to restart and the second resync point serial number equal to the sum of the checkpoint identifier, on the F-RESTART primitive, and the synchronization offset of the corresponding TN plus 1.
- [427] PDU with TN and the checkpoint identifier is sent as user data on a P-RESYNC request primitive with first resync item set to restart and the first resync point serial number equal to the sum of the checkpoint identifier, on the F-RESTART primitive, and the synchronization offset of the corresponding TN plus 1.
- [428] PDU with TN and the checkpoint identifier is sent as user data on a P-RESYNC response primitive with second resync item set to restart and the second resync point serial number equal to those received on the P-RESYNC indication.
- [429] Place Transfer-end response in TE response queue if response is not already in the queue.
- [430] Remove corresponding TE response from TE response queue and remove corresponding entry from checkpoint entry table.
- [431] Update the TN for the last transfer end indication.
- [432] Use as parameter the TN and the TN of the last transfer end confirm (or confirms) received. (*init read)

- [433] Use as parameter the TN and the TN of the last transfer end request (or requests) issued. (init write)
- [434] If the response queue is not empty, remove TE responses from its queue up to TN of confirm received on the indication.
- [435] Use as parameters the TN and the TN of the last transfer end indication (or indications) received. (*resp write)
- [436] Use as parameter the TN and the TN of the last transfer end response (or responses) issued. (*resp read)
- [438] Send PDU as user data with BTN = smaller of the BTN and the (transfer number on Cancel + SBTN) on a P-RESYNC request or response primitive with second item set to abandon..
- [439] Send PDU as user data with BTN = smaller of the BTN and the (transfer number on Cancel + SBTN) on a P-RESYNC request or response primitive with first item set to abandon..
- [440] If the DE queues are not empty, remove from the end of queues entries up to but not including the entry for TN, the smallest TN corresponding to the BTN
- [441] Reissue Read requests with TN > current TN and remove any write requests from transfer request queue (or queues).
- [444] Use (BTN received - SBTN) as the transfer number parameter.
- [445] Set user data to TN and checkpoint identifier of I-CHECK request (or response) primitive.
- [446] If there is no change in TN then use the TN and the checkpoint identifier received as parameters; otherwise use the new TN and the (final checkpoint identifier of the new TN +1) as parameters.
- [447] Clear R/W indicator.
- [449] The smaller of the BTN and the (TN received + SBTN) becomes the new BTN and the smallest TN in the checkpoint entry table corresponding to the new BTN becomes the new TN.
- [451] Use for the sync point serial number, the checkpoint received plus the offset for the TN specified in F-CHECK.
- [452] Use the TN in user data of P-SYCMCF and use for checkpoint the sync point serial number minus the offset for the TN.
- [453] Set discard indicator and set the outstanding checkpoint counter(s) for entries $\geq$ TN to zero.
- [454] Increment the checkpoint identifier expected for the TN specified in F-CHECK.
- [455] Increment the outstanding checkpoint counter for the TN specified in F-CHECK.
- [456] Decrement the outstanding checkpoint counter for the TN specified in F-CHECK according to the checkpoint number confirmed (see 15.3.2).
- [457] Set the checkpoint identifier expected for the corresponding TN to one.
- [458] Set the synchronization offset for the corresponding TN to the value of the second NSPN minus the checkpoint identifier expected.
- [459] Set the checkpoint identifier expected to the value negotiated plus one for the TN.
- [460] Discard any user data not yet delivered and set outstanding checkpoint counter(s) for entries $\geq$ TN to zero.
- [461] Use TN as the parameter.
- [462] The checkpoint identifier for the TN shall be equal to the value received on the PDU.
- [463] Reissue TE write response and remove any TE read response from TE response queue (or queues).

- [464] Add 1 to the first NSPN (sender).
- [465] Add 1 to the second NSPN (receiver).
- [466] Set the first NSPN equal to the synch point serial number negotiated with the presentation service provider.
- [467] Set the second NSPN equal to the synch point serial number negotiated with the presentation service provider.
- [468] Store the synchronization point serial number provided by the presentation service provider as the first NSPN.
- [469] Store the synchronization point serial number provided by the presentation service provider as the second NSPN.
- [470] Set the synchronization offset for the corresponding TN to the value of the first NSPN minus the checkpoint identifier expected.
- [471] For all but the first entry in the checkpoint table that has a BTN = current BTN (if any), increment the BTN by 1.

Note - TN stands for RTN for concurrent overlap read entity and WTN for concurrent overlap write.

A.4.5 Predicates - bulk data transfer

Relabel A.4.5:

A.4.5.1 Normal access (non-overlapped)

Add the following sub-clause.

A.4.5.2 Consecutive and Concurrent access

- P8: The context is not within the defined context set.
- P14: The synchronization point exceeds 999 998, exceeding the session limitation.
- P15: Discard indicator is set.
- P27: The outstanding checkpoint counter exceeds the agreed maximum.
- P400: The consecutive overlap functional unit has been negotiated.
- P401: The Symm Sync functional unit of Session has been negotiated.
- P402: The expected checkpoint identifier for the TN equals the sync point serial number received minus the offset for the TN.
- P403: The next request is marked (indicating that TN has already been updated in the checkpoint table)
- P404: Next transfer is read.
- P405: Next transfer is write.
- P406: Number in transfer queue exceeds limit set in open.
- P407: The symm resync functional unit has been negotiated.
- P408: The Data end request queue is empty.
- P409: Read indicator is set.
- P410: Transfer request queue is empty.
- P411: The data end indication queue is empty.

- P412: Last transfer end confirm TN received < last transfer end response TN issued.
- P413: Cancel read or restart read or transfer-end read.
- P414: Cancel write or restart write or transfer-end write.
- P415: Last transfer end indication TN received < last transfer end request TN issued.
- P416: TN on TRERP PDU = TN of last transfer indication received.
- P417: Cancel PDU preserved
- P418: The transfer end request queue is not empty.
- P419: Complete re-issue of transfer end requests
- P420: Complete re-issue of transfer end response.
- P421: TN > last TN in TE request queue.
- P422: Last transfer end indication TN received > TN of last item in TE response queue.
- P423: The TE response is already in the response queue and the TN of Transfer end response > TN of last TN confirm received by the initiating entity.
- P424: The write indicator is set.
- P425: On the Restart, Recover or Cancel primitive, the TN confirm (or confirms) received by initiator < the TN of last item in TE response queue (or queues)
- P426: On the Restart, Recover or Cancel primitive, the TN indication (or indications) received by the responder < the TN of last item in TE request queue (or queues)
- P427: TN of transfer end primitive = TN of last item in TE queue (or the last item removed from the queue if the TE queue is empty)
- P428: TN in TRERQ PDU $\leq$ last transfer end indication TN received.
- P429: TN > Last TE end indication + 1.
- P430: Next TE request in queue is read.
- P431: There are transfer requests with TN > current TN in the transfer request queue (or queues).
- P432: The TE request is already in a TE request queue and the TN of transfer end request > the last TN indication received by the responding entity.
- P433 Complete re-issue of transfer request.
- P434: There is no next transfer request.
- P436: Request already in queue.
- P437: TN=1
- P447: Both activity types are cleared

Note - TN stands for RTN for concurrent overlap read entity and WTN for concurrent overlap write entity.

A.4.6 Initiating state tables - bulk data transfer*Relabel A.4.6:***A.4.6.1 Normal access (non-overlapped) - Initiator***Relabel A.4.7:***A.4.6.2 Detailed entries - normal access***Add the following sub-clauses.***A.4.6.3 Consecutive access - Initiator**

STATE			Q-DXFRIDLE	Q-DREASYPD	Q-REESYMPD	Q-CANSYMPD	Q-IREADENG	Q-IRXFERENG	Q-WRTRSYMPD	Q-RES-SYMPD	Q-CANSYMPD		Q-IRWRTENG	Q-WXFERENG	Q-CANCELPD	Q-IRCESSTARTPD	Q-WRESSTARTPD	Q-IRRESSTARTPD	Q-IRWRESSTARTPD	Q-RESTART-CAN-PD	
EVENT			E	C	D	D	D	D	D	D	D	E	G	G	D	D	D	D	D	D	
I-REARQ	1	35	45			45	64	56	47			47	47	66							
I-WRTRQ	2	36	46			46	46	65	48			48	48	65							
I-DATRQ									4			4		51							
I-DAERQ									5			5		52							
DATIN						6	49	49							41		41				
DAERQ						8	50	50							41		41				
I-TREARQ							9						10								
TREARQ								11						3							
I-CANRQ				31	31		12	12	54	34	34		7				38	38	12	7	
CANRQ															13						
CANRQ							14			14			14	14	14	15		14	14	14	39
I-CANRP																16					
I-CHKRQ									17			17									
P-SYMPD									32	60	7	18	37	37	41						
P-SYMIN				29	21	12	19	42	42						41						
I-CHKRP							20	20	63												
I-RESRQ				30			21	21	59	33			22		60						
RESRP																	23	24		54	
RESRQ							25		62	26			26	26	26			43	44		
I-RESRP																					

A.4.6.4 Detailed entries - consecutive access

1: P400&(~P401 ~P407): P400&P401&P407:	L-ERRABT [420],[401],[409],[44],[412],[413],[457],[458],REARQ[2]	=>samestate =>Q-REA-SYMIN-PD
2: P400&(~P401 ~P407): P400&P401&P407: P14: ~P14:	L-ERRABT [420],[401],[410],[44],[412],[413],WRTRQ[2],[464], L-ERRABT P-SYMRQ,[457],[470]	=>samestate =>samestate =>Q-WRT-SYMCF-PD
3: P15: ~P15: P410: ~P410&P409: ~P410&P424&P408&P427: ~P410&P424&~P408&P427: ~P410&~P430&~P427:	I-TRECF,[405],[406], [412],[411],[457],[458]	=>samestate =>Q-DXFRIDLE =>Q-READ =>Q-WRITE =>Q-I-WRT-ENDING =>Q-W-XFER-ENDING
4: P8: ~P8:	L-ERRABT P-DATRQ[18]	=>samestate =>samestate
5:	DAERQ[16],[407]	=>Q-I-WRT-ENDING
6: P15: ~P15:	I-DATIN	=>samestate =>samestate
7:	[453],[460],CANRQ[417]	=>Q-CANCEL-PD
8: P15: ~P15: ~P404&~P405: P404&~P403: P404&P403: P405:	I-DAEIN,[414], [447] [412],[411],[409],[457],[458] [409] [410]	=>samestate =>Q-I-READ-ENDING =>Q-I-READ-ENDING =>Q-I-READ-ENDING =>Q-I-READ-ENDING
9: P432: ~P432: P404&~P403: P404&P403: P405: ~P404&~P405:	TRERQ[402] [415],TRERQ[432],[402],[404],[471], [44],[409],[412],[411],[457],[458] [44],[409] [44],[410],[412],[411],[457],[470] [447]	=>samestate =>Q-R-XFER-ENDING =>Q-R-XFER-ENDING =>Q-R-XFER-ENDING =>Q-R-XFER-ENDING
10: P404: P405&~P403: P405&P403: ~P404&~P405:	[408],TRERQ[402],[432],[404],[471], [44],[409] [44],[410],[412],[411],[457],[470] [44],[410] [447]	=>Q-W-XFER-ENDING =>Q-W-XFER-ENDING =>Q-W-XFER-ENDING =>Q-W-XFER-ENDING
11: P15: ~P15: P410&P427: P410&~P427: ~P410&P409&P411&P427: ~P410&P424&P408&P427: ~P410&P424&~P408&P427: ~P410&P409&~P411&P427: ~P410&~P427&P430 ~P410&~P427&~P430	I-TRECF,[405],[406],	=>samestate =>Q-DXFRIDLE =>samestate =>Q-READ =>Q-WRITE =>Q-I-WRT-ENDING =>Q-I-READ-ENDING =>Q-R-XFER-ENDING =>Q-W-XFER-ENDING

12:	[453],[460],CANRQ[416]	=>Q-CANCEL-PD
13: ~P414&P413:	[449],[469],[21],I-CANCF[461],[440]	=>Q-DXFRIDLE
P414&~P413:	[449],[468],[21],I-CANCF[461],[440]	=>Q-DXFRIDLE
14: ~P414&P413:	[449],[469],[453],[460],I-CANIN[461]	=>Q-I-CANCEL-PD
P414&~P413:	[449],[468],[453],[460],I-CANIN[461]	=>Q-I-CANCEL-PD
15: ~P413&P414:	[449],CANRP[417],[468],[21],I-CANCF[461],[440]	=>Q-XFRIDLE
~P414&P413:	[449],CANRP[416],[469],[21],I-CANCF[461],[440]	=>Q-XFRIDLE
16: ~P413&P414:	[468],[21],CANRP[417],[440]	=>Q-XFRIDLE
~P414&P413:	[469],[21],CANRP[416],[440]	=>Q-XFRIDLE
17:	[25],[454],[464],	
~P14:	P-SYMRQ[445]	=>samestate
P14:	L-ERRABT	=>samestate
18:	I-CHKCF[452]	=>samestate
19: ~P27&P402:	I-CHKIN[403],[465],[454]	=>samestate
P27 ~P402:	L-ERRABT	=>samestate
20:	P-SYMRP[451][445]	=>samestate
21:	[453],RESRQ[426]	=>Q-RRESTART-PD
22:	[453],RESRQ[427]	=>Q-WRESTART-PD
23:	[449],[467],[459],[458],[21],	
P418:	I-RESCF[446]	=>Q-R-XFER-ENDING
~P418:	I-RESCF[446]	=>Q-READ
24:	[449],[466],[459],[470],[21],	
P418:	I-RESCF[446]	=>Q-W-XFER-ENDING
~P418:	I-RESCF[446]	=>Q-WRITE
25:	[449],I-RESIN[446],[460]	=>Q-I-RRESTART-PD
26:	[449],I-RESIN[446]	=>Q-I-WRESTART-PD
27: P418:	[467],[459],[458],RESRP[428],[440]	=>Q-R-XFER-ENDING
~P418&P411:	[467],[459],[458],RESRP[428],[440]	=>Q-READ-ENDING
~P418&~P411:	[467],[459],[458],RESRP[428],[440]	=>Q-READ
28: P418:	[466],[459],[470],RESRP[425],[440]	=>Q-W-XFER-ENDING
~P418&P408:	[466],[459],[470],RESRP[425],[440]	=>Q-I-WRT-ENDING
~P418&~P408:	[466],[459],[470],RESRP[425],[440]	=>Q-WRITE
29:	[465],[458],P-SYMRP	=>Q-READ
30:	[453],[47]	=>Q-RES-SYMIN-PD
31:	[453],[47]	=>Q-CAN-SYMIN-PD
32:	[470]	=>Q-WRITE
33:	[453],[47]	=>Q-RES-SYMCF-PD
34:	[453],[47]	=>Q-CAN-SYMCF-PD
35:	REARQ[2],[458],[409]	=>Q-READ-SYMIN-PD