```
-- file: FSPDefs.Mesa; edited by Sandman on March 22, 1978  3:23 PM

DIRECTORY
  AltoDefs: FROM "altodefs";

DEFINITIONS FROM AltoDefs;

FSPDefs: DEFINITIONS =
  BEGIN

  -- types and formats of zone headers and nodes

  BlockSize: TYPE = CARDINAL [0..77777B--VMLimit/2--];

  NodePointer: TYPE = POINTER TO NodeHeader;
    FreeNodePointer: TYPE = POINTER TO free NodeHeader;

  NodeHeader: TYPE = PRIVATE RECORD [
    length: BlockSize,
    extension: SELECT state: * FROM
      inuse =>
        NULL,
      free => [
        fwdp, backp: FreeNodePointer],
      ENDCASE];


  Deallocator: TYPE = PROCEDURE [POINTER];

  ZonePointer: TYPE = POINTER TO ZoneHeader;

  ZoneHeader: TYPE = PRIVATE RECORD [
    node: free NodeHeader,
    rover: FreeNodePointer,
      -- roving pointer to slow down fragmentation
      -- (see Knuth, Vol I, p. 597 #6)
    lock: MONITORLOCK,
    restOfZone: ZonePointer,  -- link to additional segments of zone
    length: BlockSize,
    deallocate: Deallocator,
    threshold: PUBLIC BlockSize,
    checking: PUBLIC BOOLEAN];

  ZoneOverhead: CARDINAL = SIZE[ZoneHeader]+SIZE[inuse NodeHeader];
  NodeOverhead: CARDINAL = SIZE[inuse NodeHeader];
  -- NOTE: A zone whose largest possible node is N words, must have
  --   N + ZoneOverhead + NodeOverhead words of storage

  -- public procedures and signals

  MakeNewZone: PROCEDURE [base: POINTER, length: BlockSize, deallocate: Deallocator]
    RETURNS [z: ZonePointer];
  MakeZone: PROCEDURE [base: POINTER, length: BlockSize] RETURNS [z: ZonePointer];
  AddToNewZone: PROCEDURE
    [z: ZonePointer, base: POINTER, length: BlockSize, deallocate: Deallocator];
  AddToZone: PROCEDURE [z: ZonePointer, base: POINTER, length: BlockSize];
  PruneZone: PROCEDURE [z: ZonePointer] RETURNS [BOOLEAN];
  DestroyZone: PROCEDURE [z: ZonePointer];
  DoNothingDeallocate: Deallocator;  -- adds storage to System Heap

  NoRoomInZone: SIGNAL [z: ZonePointer];  -- not enough space to fill a request

  MakeNode: PROCEDURE [z: ZonePointer, n: BlockSize] RETURNS [POINTER];
  FreeNode: PROCEDURE [z: ZonePointer, p: POINTER];
  SplitNode: PROCEDURE [z: ZonePointer, p: POINTER, n: BlockSize];
  NodeSize: PROCEDURE [p: POINTER] RETURNS [BlockSize];

  ZoneTooSmall: ERROR [POINTER];
  InvalidZone: ERROR [POINTER];  -- zone header looks fishy
  NodeLoop: ERROR [ZonePointer];
  InvalidNode: ERROR [POINTER];  -- node appears damaged

  END.
```