

CCSDS Historical Document

This document's Historical status indicates that it is no longer current. It has either been replaced by a newer issue or withdrawn because it was deemed obsolete. Current CCSDS publications are maintained at the following location:

<http://public.ccsds.org/publications/>

***Consultative
Committee for
Space Data Systems***

RECOMMENDATION FOR SPACE
DATA SYSTEMS STANDARDS

**ADVANCED ORBITING SYSTEMS,
NETWORKS AND DATA LINKS:**

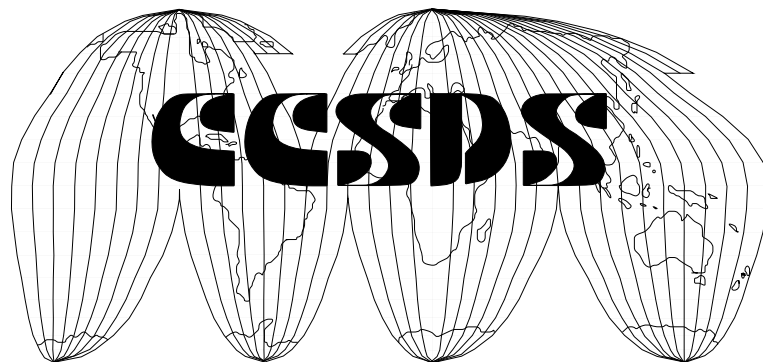
ABSTRACT DATA TYPE LIBRARY

ADDENDUM TO CCSDS 701.0-B-2

CCSDS 705.1-B-1

BLUE BOOK

May 1994



CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

AUTHORITY

Issue:	Blue Book, Issue 1
Date:	May 1994
Location:	Villafranca, Spain

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS Recommendations is detailed in reference [1], and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code OI)
National Aeronautics and Space Administration
Washington, DC 20546, USA

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of member space Agencies. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed RECOMMENDATIONS and are not considered binding on any Agency.

This RECOMMENDATION is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of this RECOMMENDATION is entirely voluntary. Endorsement, however, indicates the following understandings:

- o Whenever an Agency establishes a CCSDS-related STANDARD, this STANDARD will be in accord with the relevant RECOMMENDATION. Establishing such a STANDARD does not preclude other provisions which an Agency may develop.
- o Whenever an Agency establishes a CCSDS-related STANDARD, the Agency will provide other CCSDS member Agencies with the following information:
 - The STANDARD itself.
 - The anticipated date of initial operational capability.
 - The anticipated duration of operational service.
- o Specific service arrangements shall be made via memoranda of agreement. Neither this RECOMMENDATION nor any ensuing STANDARD is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this Recommendation will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or, (3) be retired or cancelled.

In those instances when a new version of a RECOMMENDATION is issued, existing CCSDS-related Agency standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each Agency to determine when such standards or implementations are to be modified. Each Agency is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommendation.

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

FOREWORD

This document, which is a technical Recommendation prepared by the Consultative Committee for Space Data Systems (CCSDS), is intended for use by participating space Agencies in their development of 'Advanced Orbiting Systems'.

This Recommendation, written using the ISO Formal Description Technique LOTOS, contains a library of Abstract Data Types used by the formal specifications contained in references [4], [5], and [6].

An overview of the CCSDS Validation Programme (of which this document is an output) may be found in reference [7].

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommendation is therefore subject to CCSDS document management and change control procedures which are defined in reference [1].

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

At time of publication, the active Member and Observer Agencies of the CCSDS were

Member Agencies

- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA HQ)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.

Observer Agencies

- Australian Space Office (ASO)/Australia.
- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (SPO)/Belgium.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Industry Canada/Communications Research Center (CRC)/Canada.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 705.1-B-1	Recommendation for Space Data Systems Standards—Advanced Orbiting Systems, Networks and Data Links: Abstract Data Type Library—Addendum to CCSDS 701.0-B-2, Issue 1	May 1994	Original Issue

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

CONTENTS

<u>Sections</u>	<u>Page</u>
REFERENCES	vii
1 PURPOSE AND SCOPE	1-1
2 PATH DATA TYPES	2-1
2.1 PATH SERVICE DATA TYPES	2-1
2.2 PATH PROTOCOL DATA TYPES	2-4
2.3 PATH SUBNETWORK DATA TYPES	2-25
3 VCLC DATA TYPES	3-1
3.1 VCLC SERVICE DATA TYPES	3-1
3.2 VCLC PROTOCOL DATA TYPES	3-3
4 VCA DATA TYPES	4-1
4.1 VCA SERVICE DATA TYPES	4-1
4.2 VCA PROTOCOL DATA TYPES	4-2
5 LOTOS DATA TYPES	5-1
5.1 BOOLEAN DEFINITION	5-1
5.2 BASIC NATURAL NUMBER TYPE	5-2
5.3 NATURAL NUMBER	5-3
5.4 BIT	5-4
5.5 OCTET	5-5
5.6 BITSTRING	5-7
5.7 OCTETSTRING	5-9

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

REFERENCES

- [1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-6. Yellow Book. Issue 6. Washington, D.C.: CCSDS, May 1994 or later issue.
- [2] *Advanced Orbiting Systems, Networks and Data Links: Architectural Specification*. Recommendation for Space Data Systems Standards, CCSDS 701.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 1992 or later issue.
- [3] *Information Processing Systems—Open Systems Interconnection—LOTOS—A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*. ISO 8807. Issue 1. Geneva: ISO, 1989.
- [4] *Advanced Orbiting Systems, Networks and Data Links: Formal Specification of the Path Service and Protocol—Addendum to CCSDS 701.0-B-2*. Recommendation for Space Data Systems Standards, CCSDS 705.2-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1994 or later issue.
- [5] *Advanced Orbiting Systems, Networks and Data Links: Formal Specification of the VCLC Service and Protocol—Addendum to CCSDS 701.0-B-2*. Recommendation for Space Data Systems Standards, CCSDS 705.3-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1994 or later issue.
- [6] *Advanced Orbiting Systems, Networks and Data Links: Formal Specification of the VCA Service and Protocol—Addendum to CCSDS 701.0-B-2*. Recommendation for Space Data Systems Standards, CCSDS 705.4-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1994 or later issue.
- [7] *Advanced Orbiting Systems, Networks and Data Links: Formal Definition of CPN Protocols, Methodology and Approach*. Report Concerning Space Data Systems Standards, CCSDS 705.0-G-2. Green Book. Issue 2. Washington, D.C.: CCSDS, October 1993 or later issue.
- [8] *Advanced Orbiting Systems, Networks and Data Links: Summary of Concept, Rationale and Performance*. Report Concerning Space Data Systems Standards, CCSDS 700.0-G-3. Green Book. Issue 3. Washington, D.C.: CCSDS, November 1992 or later issue.

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

1 PURPOSE AND SCOPE

This document provides an Abstract Data Type Library for use in formal specifications of Consultative Committee for Space Data Systems (CCSDS) Advanced Orbiting Systems (AOS) services and protocols¹ using the ISO LOTOS formal description technique (refer to reference [3]). These formal specifications are not intended as replacements for the natural-language specifications provided in the AOS Blue Book (reference [2]), but as unambiguous expressions of those specifications, which may be used to clarify any problem areas.

This document is one of four CCSDS Recommendations that provide LOTOS specifications for the suite of AOS services and protocols (see references [4] through [6]). The relationship between the main AOS Recommendation and the four LOTOS Specifications is shown below; the numbers to the right are the CCSDS document references for the Recommendations containing the LOTOS Specifications.

ADT Library	705.1
Path Service	705.2
Path Protocol	705.2
VCLC Service	705.3
VCLC Protocol	705.3
VCA Service	705.4
VCA Protocol	705.4

A supporting CCSDS Report (reference [7]) contains the rationale, methodology, and approach used to prepare the LOTOS specifications.

These documents are expected to be of use primarily to the technical experts responsible for the design, configuration, and testing of AOS implementations; a basic knowledge of LOTOS is required to understand the formal specifications. Other users of the AOS services should consult the main AOS Recommendation and the companion CCSDS Report (references [2] and [8]).

¹The natural-language specifications for the AOS services and protocols are contained in the AOS Blue Book, CCSDS 701.0-B-2, reference [2].

2 PATH DATA TYPES

2.1 PATH SERVICE DATA TYPES

2.1.1 Octet Service Data Unit

The Octet Service Data Unit (O_SDU) is a delimited, octet-oriented data unit whose content and format are unknown to the Path layer. As such, the 'OctetString' ADT (5.7) is identical in function and will be used instead of a special O_SDU ADT.

2.1.2 Packet Service Data Unit

The Packet Service Data Unit (P_SDU) is a Version-1 CCSDS Packet that has been created by the service user. As such, a separate ADT is not needed, and the CCSDSPacket ADT (2.2.1) will be used instead.

2.1.3 Application Process Identifier Qualifier

The Application Process Identifier (APID) Qualifier is an optional parameter which is associated with the APID in the CCSDS Packet in order to maintain global uniqueness of APIDs. The APID Qualifier is of unknown format and length, and hence in this definition it is taken to be a string of bits added to a base creator type. Two APIDQualifiers are equal only if they are of the same length and have the same contents.

```

type      APIDQualifier is Bit, Boolean
sorts    APIDQual
opns     Add                      : Bit, APIDQual -> APIDQual
         NullAPIDQual            : -> APIDQual
         _Eq_, _Ne_              : APIDQual, APIDQual -> Bool
eqns     forall b1, b2 : Bit, QUAL1, QUAL2 : APIDQual

         ofsort Bool

         NullAPIDQual Eq NullAPIDQual = True ;

         NullAPIDQual Eq Add(b1, QUAL1) = False ;

         Add(b1, QUAL1) Eq NullAPIDQual = False ;

         Add(b1, QUAL1) Eq Add(b2, QUAL2) =
           (b1 Eq b2) And (QUAL1 Eq QUAL2) ;

         QUAL1 Ne QUAL2 = Not(QUAL1 Eq QUAL2) ;

endtype

```

2.1.4 Secondary Header Indicator

Within the Octet String service, the Secondary Header Indicator parameter signals the presence of a Secondary Header at the start of the O_SDU. The Octet String service parameter is distinguished from the Secondary Header Flag within the Packet service.

```

type      SecondaryHeaderIndicator is Boolean, SecondaryHeaderFlag
sorts     SecondaryHeaderIndicator
opns      Absent                    : -> SecondaryHeaderIndicator
          Present                   : -> SecondaryHeaderIndicator
          SHToSHF                   : SecondaryHeaderIndicator -> SHF
          SHFToS                      : SHF -> SecondaryHeaderIndicator

eqns

ofsort SHF

SHToSHF(Absent) = SHF(0) ;
SHToSHF(Present) = SHF(1) ;

ofsort SecondaryHeaderIndicator

SHFToS(SHF(0)) = Absent ;
SHFToS(SHF(1)) = Present ;

endtype

```

2.1.5 Path ID

The Path ID, which uniquely identifies the Logical Data Path (LDP), consists of the APID plus the optional APID Qualifier. Operations are provided to extract the two parts of the Path ID and to compare Path IDs.

```

type      PathID is APID, APIDQualifier, Boolean
sorts     PathID
opns      MakePathID                : APID, APIDQual -> PathID
          QualifierPart              : PathID -> APIDQual
          APIDPart                   : PathID -> APID
          _Ne_                        : PathID, PathID -> Bool
          _Eq_                        : PathID, PathID -> Bool

eqns

forall APID1 : APID, QUAL1 : APIDQual, PT1, PT2 : PathID

ofsort APID

APIDPart(MakePathID(APID1, QUAL1)) = APID1 ;

ofsort APIDQual

QualifierPart(MakePathID(APID1, QUAL1)) = QUAL1 ;

ofsort Bool

PT1 Eq PT2 = (APIDPart(PT1) Eq APIDPart(PT2)) And
             (QualifierPart(PT1) Eq QualifierPart(PT2)) ;

PT1 Ne PT2 = (APIDPart(PT1) Ne APIDPart(PT2)) Or
             (QualifierPart(PT1) Ne QualifierPart(PT2)) ;

endtype

```

2.1.6 Octet String Data Loss Indicator

The Data Loss Indicator is used to alert the user of the Octet String service in a destination end system that one or more O_SDUs have been lost during transmission, as evidenced by a discontinuity in the CCSDS Path Protocol Data Unit (CP_PDU) Sequence Count. This is an optional parameter, the presence or absence of which is implementation specific.

```

type      DataLossIndicator is Boolean
sorts    DataLossIndicator
opns     OSDULost           : -> DataLossIndicator
         OSDUNotLost       : -> DataLossIndicator
         _Eq_              : DataLossIndicator,
                           DataLossIndicator -> Bool

eqns     forall DLI1, DLI2 : DataLossIndicator

         ofsort Bool

         OSDULost Eq OSDULost      = true ;
         OSDUNotLost Eq OSDUNotLost = true ;
         OSDULost Eq OSDUNotLost   = false ;
         OSDUNotLost Eq OSDULost    = false ;
endtype

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.2 PATH PROTOCOL DATA TYPES

The CCSDS Path Protocol Data Unit (CP_PDU) is the Version-1 CCSDS Packet.

2.2.1 CCSDS Packet

The type definition for the Version-1 CCSDS packet is built from the Primary Header (2.2.2) and User Data (2.2.3) ADTs. Since the Optional Secondary Header is not used by the protocols being specified, it is considered part of the UserData.

```
type CCSDSPacket is PrimaryHeader, OctetString, FillData, NaturalNumber,
                    PacketType, Boolean
sorts CCSDSPacket
opns  MakeCCSDSPacket      : PrimaryHeader, OctetString -> CCSDSPacket
      GetPrimaryHeader    : CCSDSPacket -> PrimaryHeader
      GetUserData         : CCSDSPacket -> OctetString
      TotalLengthofPacket : CCSDSPacket -> Nat
      ConvertPkttoOS      : CCSDSPacket -> OctetString
      ConvertOSToPkt      : OctetString -> CCSDSPacket
      MakeFillPacket      : PacketType, Nat, OctetString -> CCSDSPacket
      ValidPacketLength   : CCSDSPacket -> Bool

eqns forall PH : PrimaryHeader,
         UD, OS, FillPattern : OctetString,
         FillLength : Nat,
         TYP : PacketType,
         CP1 : CCSDSPacket

  ofsort PrimaryHeader

  GetPrimaryHeader(MakeCCSDSPacket(PH, UD)) = PH ;

  ofsort OctetString

  GetUserData(MakeCCSDSPacket(PH, UD)) = UD ;

  ConvertPkttoOS(MakeCCSDSPacket(PH, UD)) = Append(UD, ConvertPHtoOS(PH)) ;

  ofsort Nat

  TotalLengthofPacket(MakeCCSDSPacket(PH, UD)) = LengthOf(UD) + 6;

  ofsort CCSDSPacket

  ConvertOSToPkt(OS) = MakeCCSDSPacket(ConvertOSToPH(RetainOctets(OS,6)),
                                       StripOctets(OS,6)) ;

  MakeFillPacket(TYP,
                 FillLength,
                 FillPattern) = MakeCCSDSPacket(
                               MakeFillPH(TYP, FillLength),
                               MakeFillData(FillPattern, FillLength)) ;

  ofsort Bool

  ValidPacketLength(
  MakeCCSDSPacket(PH, UD) ) = TotalLengthofPacket(
                               MakeCCSDSPacket(PH, UD) )
  Eq
  HeaderIndicatedLengthOfPacket( PH ) ;

endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.2.2 Primary Header

The type definition for the Primary Header is made up from the Packet ID (2.2.2.1), Sequence Control (2.2.2.2) and Packet Length (2.2.2.3) ADTs. Extra definitions are provided to extract component ADTs from the complete Primary Header.

```
type PrimaryHeader is Version, PacketType, SecondaryHeaderFlag,
    APID, SequenceFlags, PacketSequenceCount,
    PacketLength, PacketID, PacketSequenceControl,
    NaturalNumber, OctetString

sorts PrimaryHeader
opns  GetVersion          : PrimaryHeader -> Version
      GetPacketType      : PrimaryHeader -> PacketType
      GetSHF             : PrimaryHeader -> SHF
      GetAPID           : PrimaryHeader -> APID
      GetSequenceFlags  : PrimaryHeader -> SequenceFlags
      GetPacketSequenceCount : PrimaryHeader -> PacketSequenceCount
      GetPacketLength   : PrimaryHeader -> PacketLength
      MakePrimaryHeader : PacketID,
                          PacketSC,
                          PacketLength -> PrimaryHeader
      ConvertPHtoOS     : PrimaryHeader -> OctetString
      ConvertOStoPH     : OctetString -> PrimaryHeader
      MakeFillPH        : PacketType, Nat -> PrimaryHeader
      LengthOfPacketInOS : OctetString -> Nat
      HeaderIndicatedLengthOfPacket : PrimaryHeader -> Nat

eqns  forall VERS : Version,
      PKT, TYP : PacketType,
      SHF : SHF,
      APID : APID,
      SEQFLAG : SequenceFlags,
      SEQCOUNT : PacketSequenceCount,
      PKTLENGTH : PacketLength,
      PID : PacketID,
      PSC : PacketSC,
      OS : OctetString,
      PH : PrimaryHeader,
      Nat1 : Nat

      ofsort Version

      GetVersion(MakePrimaryHeader(MakePacketID(VERS, PKT, SHF, APID),
        MakePacketSC(SEQFLAG, SEQCOUNT), PKTLENGTH)) = VERS ;

      ofsort PacketType

      GetPacketType(MakePrimaryHeader(MakePacketID(VERS, PKT, SHF, APID),
        MakePacketSC(SEQFLAG, SEQCOUNT), PKTLENGTH)) = PKT ;

      ofsort SHF

      GetSHF(MakePrimaryHeader(MakePacketID(VERS, PKT, SHF, APID),
        MakePacketSC(SEQFLAG, SEQCOUNT), PKTLENGTH)) = SHF ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

ofsort APID

```
GetAPID(MakePrimaryHeader(MakePacketID(VERS, PKT, SHF, APID),  
    MakePacketSC(SEQFLAG, SEQCOUNT), PKTLENGTH)) = APID ;
```

ofsort SequenceFlags

```
GetSequenceFlags(MakePrimaryHeader(MakePacketID(VERS, PKT, SHF, APID),  
    MakePacketSC(SEQFLAG, SEQCOUNT), PKTLENGTH)) = SEQFLAG ;
```

ofsort PacketSequenceCount

```
GetPacketSequenceCount(MakePrimaryHeader(  
    MakePacketID(VERS, PKT, SHF, APID),  
    MakePacketSC(SEQFLAG, SEQCOUNT),  
    PKTLENGTH)) = SEQCOUNT ;
```

ofsort PacketLength

```
GetPacketLength(MakePrimaryHeader(MakePacketID(VERS, PKT, SHF, APID),  
    MakePacketSC(SEQFLAG, SEQCOUNT),  
    PKTLENGTH)) = PKTLENGTH ;
```

ofsort OctetString

```
ConvertPHtoOS(  
    MakePrimaryHeader(  
        MakePacketID(VERS, PKT, SHF, APID),  
        MakePacketSC(SEQFLAG, SEQCOUNT),  
        PKTLENGTH)) = AddFront(  
    Octet(  
        Bit1(VERS),  
        Bit2(VERS),  
        Bit3(VERS),  
        Bit1(PKT),  
        Bit1(SHF),  
        Bit1(APID),  
        Bit2(APID),  
        Bit3(APID)),  
    AddFront(  
    Octet(  
        Bit4(APID),  
        Bit5(APID),  
        Bit6(APID),  
        Bit7(APID),  
        Bit8(APID),  
        Bit9(APID),  
        Bit10(APID),  
        Bit11(APID)),  
    AddFront(  
    Octet(  
        Bit1(SEQFLAG),  
        Bit2(SEQFLAG),  
        Bit1(SEQCOUNT),  
        Bit2(SEQCOUNT),  
        Bit3(SEQCOUNT),  
        Bit4(SEQCOUNT),  
        Bit5(SEQCOUNT),  
        Bit6(SEQCOUNT)),
```


CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
AddFront(
  Octet(
    Bit7(SEQCOUNT),
    Bit8(SEQCOUNT),
    Bit9(SEQCOUNT),
    Bit10(SEQCOUNT),
    Bit11(SEQCOUNT),
    Bit12(SEQCOUNT),
    Bit13(SEQCOUNT),
    Bit14(SEQCOUNT)),
  AddFront(
    Octet(
      Bit1(PKTLLENGTH),
      Bit2(PKTLLENGTH),
      Bit3(PKTLLENGTH),
      Bit4(PKTLLENGTH),
      Bit5(PKTLLENGTH),
      Bit6(PKTLLENGTH),
      Bit7(PKTLLENGTH),
      Bit8(PKTLLENGTH)),
    AddFront(
      Octet(
        Bit9(PKTLLENGTH),
        Bit10(PKTLLENGTH),
        Bit11(PKTLLENGTH),
        Bit12(PKTLLENGTH),
        Bit13(PKTLLENGTH),
        Bit14(PKTLLENGTH),
        Bit15(PKTLLENGTH),
        Bit16(PKTLLENGTH)),
      NullOS)))));
```

ofsort PrimaryHeader

```
ConvertOStoPH(OS) = MakePrimaryHeader(
  MakePacketID(
    Version(Bit1(Nth(OS,1)),
            Bit2(Nth(OS,1)),
            Bit3(Nth(OS,1))),
    PacketType(Bit4(Nth(OS,1))),
    SHF(Bit5(Nth(OS,1))),
    APID(Bit6(Nth(OS,1)),
          Bit7(Nth(OS,1)),
          Bit8(Nth(OS,1)),
          Bit1(Nth(OS,2)),
          Bit2(Nth(OS,2)),
          Bit3(Nth(OS,2)),
          Bit4(Nth(OS,2)),
          Bit5(Nth(OS,2)),
          Bit6(Nth(OS,2)),
          Bit7(Nth(OS,2)),
          Bit8(Nth(OS,2))
    ),
  MakePacketSC(
    SequenceFlags(
      Bit1(Nth(OS,3)),
      Bit2(Nth(OS,3))),
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
PacketSequenceCount(  
    Bit3(Nth(OS,3)),  
    Bit4(Nth(OS,3)),  
    Bit5(Nth(OS,3)),  
    Bit6(Nth(OS,3)),  
    Bit7(Nth(OS,3)),  
    Bit8(Nth(OS,3)),  
    Bit1(Nth(OS,4)),  
    Bit2(Nth(OS,4)),  
    Bit3(Nth(OS,4)),  
    Bit4(Nth(OS,4)),  
    Bit5(Nth(OS,4)),  
    Bit6(Nth(OS,4)),  
    Bit7(Nth(OS,4)),  
    Bit8(Nth(OS,4))  
),  
PacketLength(  
    Bit1(Nth(OS,5)),  
    Bit2(Nth(OS,5)),  
    Bit3(Nth(OS,5)),  
    Bit4(Nth(OS,5)),  
    Bit5(Nth(OS,5)),  
    Bit6(Nth(OS,5)),  
    Bit7(Nth(OS,5)),  
    Bit8(Nth(OS,5)),  
    Bit1(Nth(OS,6)),  
    Bit2(Nth(OS,6)),  
    Bit3(Nth(OS,6)),  
    Bit4(Nth(OS,6)),  
    Bit5(Nth(OS,6)),  
    Bit6(Nth(OS,6)),  
    Bit7(Nth(OS,6)),  
    Bit8(Nth(OS,6))  
)  
);  
  
MakeFillPH(TYP, Nat1) = MakePrimaryHeader(  
    MakePacketID(  
        Version1,  
        TYP,  
        SHAbsent,  
        FillPacketAPID),  
    MakePacketSC(  
        PacketSequenceUnSeg,  
        PacketSequenceCount(0,0,0,0,0,0,0,  
                             0,0,0,0,0,0,0)),  
    ConvertNattoPL(Pred(Nat1)) );  
  
ofsort Nat  
  
HeaderIndicatedLengthOfPacket(PH) = (ConvertPLToNat(  
    GetPacketLength(PH)) + 7);  
  
LengthOfPacketInOS(OS) = (HeaderIndicatedLengthOfPacket  
    (ConvertOStoPH(RetainOctets(OS,6))));  
  
endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.2.2.1 Packet ID

The Packet Identification (Packet ID) is made up of the following components:

- Version (PacketVersion) (2.2.2.1.1);
- Type (PacketType) (2.2.2.1.2);
- Secondary Header Flag (PacketSecondaryHeaderFlag) (2.2.2.1.3); and
- Application Process Identification (PacketAPID) (2.2.2.1.4).

The size and order of these components is as follows:

<u>PacketID Components</u>	<u>Field Length in Bits</u>
PacketVersion	3
PacketType	1
PacketSecondaryHeaderFlag	1
PacketAPID	11

Special functions are defined for obtaining the values of Packet ID components.

```
type    PacketID is Version, PacketType, SecondaryHeaderFlag, APID
sorts   PacketID
opns    MakePacketID           : Version, PacketType,
                                SHF, APID -> PacketID
endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.2.2.1.1 Version Type

Version is the version number of the CCSDS packet. The value of Version is a constant (000) indicating the Version-1 CCSDS Packet.

```
type      Version is Bit, Boolean
sorts    Version
opns     Version      : Bit, Bit, Bit -> Version
         Version1    : -> Version
         Bit1, Bit2, Bit3 : Version -> Bit
         _Eq_, _Ne_    : Version, Version -> Bool

eqns     forall b1, b2, b3, b4, b5, b6 : Bit, VERS1, VERS2 : Version

ofsort Version

Version1 = Version(0,0,0) ;

ofsort Bit

Bit1(Version(b1, b2, b3))      = b1 ;
Bit2(Version(b1, b2, b3))      = b2 ;
Bit3(Version(b1, b2, b3))      = b3 ;

ofsort Bool

Version(b1,b2,b3) Eq Version(b4,b5,b6) =
  (b1 Eq b4) And (b2 Eq b5) And (b3 Eq b6) ;

VERS1 Ne VERS2                  = Not(VERS1 Eq VERS2) ;

endtype
```

2.2.2.1.2 Type

```

type      PacketType is Bit, Boolean
sorts    PacketType
opns     Bit1                : PacketType -> Bit
         PacketType          : Bit -> PacketType
         _Eq_, _Ne_         : PacketType, PacketType -> Bool

eqns     forall b1 : Bit, PT1, PT2 : PacketType

         ofsort Bit

         Bit1(PacketType(b1)) = b1 ;

         ofsort Bool

         PT1 Eq PT2           = Bit1(PT1) Eq Bit1(PT2) ;

         PT1 Ne PT2          = Not(PT1 Eq PT2) ;

endtype

```

2.2.2.1.3 Secondary Header Flag

The Secondary Header Flag indicates the presence or absence of a Secondary Header in the CP_PDU. Constants are defined for the present (0) and absent (1) values.

```

type      SecondaryHeaderFlag is Bit, Boolean
sorts    SHF
opns     Bit1                : SHF -> Bit
         SHF                 : Bit -> SHF
         SHPresent           : -> SHF
         SHAbsent            : -> SHF
         _Eq_, _Ne_          : SHF, SHF -> Bool
eqns     forall b1, b2 : Bit, SHF1, SHF2 : SHF

         ofsort SHF

         SHPresent = SHF(1) ;

         SHAbsent = SHF(0) ;

         ofsort Bit

         Bit1(SHF(b1))      = b1 ;

         ofsort Bool

         SHF(b1) Eq SHF(b2) = b1 Eq b2 ;
         SHF(b1) Ne SHF(b2) = b1 Ne b2 ;

endtype

```

2.2.2.1.4 Application Process Identifier (APID)

The APID (possibly in conjunction with the optional external APID Qualifier) provides the naming mechanism for the LDP. Constants are defined for Fill Packet APID (2047), Encapsulated 8473 Packet APID (2046). A check is also defined for User APIDs (0-2031).

```

type      APID is Bit, Boolean
sorts     APID
opns      APID                : Bit, Bit, Bit, Bit, Bit, Bit,
                               Bit, Bit, Bit, Bit, Bit -> APID

      Bit1, Bit2, Bit3, Bit4,
      Bit5, Bit6, Bit7, Bit8,
      Bit9, Bit10, Bit11      : APID -> Bit
      UserAPID                 : APID -> Bool
      FillPacketAPID           : -> APID
      8473EncapPacketAPID      : -> APID
      _Eq_, _Ne_               : APID, APID -> Bool

eqns      forall b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11 : Bit,
                               APID1, APID2 : APID

ofsort APID

FillPacketAPID = APID(1,1,1,1,1,1,1,1,1,1,1,1) ;

8473EncapPacketAPID = APID(1,1,1,1,1,1,1,1,1,1,1,0) ;

ofsort Bit

Bit1(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b1 ;

Bit2(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b2 ;

Bit3(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b3 ;

Bit4(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b4 ;

Bit5(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b5 ;

Bit6(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b6 ;

Bit7(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b7 ;

Bit8(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b8 ;

Bit9(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b9 ;

Bit10(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b10 ;

Bit11(APID(b1, b2, b3, b4, b5, b6,
          b7, b8, b9, b10, b11))           = b11 ;

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

ofsort Bool

```
APID1 Eq APID2 = (Bit1(APID1) Eq Bit1(APID2)) And
                (Bit2(APID1) Eq Bit2(APID2)) And
                (Bit3(APID1) Eq Bit3(APID2)) And
                (Bit4(APID1) Eq Bit4(APID2)) And
                (Bit5(APID1) Eq Bit5(APID2)) And
                (Bit6(APID1) Eq Bit6(APID2)) And
                (Bit7(APID1) Eq Bit7(APID2)) And
                (Bit8(APID1) Eq Bit8(APID2)) And
                (Bit9(APID1) Eq Bit9(APID2)) And
                (Bit10(APID1) Eq Bit10(APID2)) And
                (Bit11(APID1) Eq Bit11(APID2)) ;
```

```
APID1 Ne APID2 = Not(APID1 Eq APID2) ;
```

```
UserAPID(Apid(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) =
    (b1 Eq 0) Or (b2 Eq 0) Or (b3 Eq 0) Or (b4 Eq 0) Or
    (b5 Eq 0) Or (b6 Eq 0) Or (b7 Eq 0) ;
```

endtype

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.2.2.2 Packet Sequence Control

The Packet Sequence Control type contains the Sequence Flags and the Packet Sequence Count.

<u>PacketSequenceControl Components</u>	<u>Field Length in Bits</u>
SequenceFlags	2
PacketSequenceCount	14

```
type PacketSequenceControl is SequenceFlags, PacketSequenceCount
sorts PacketSC
opns  MakePacketSC          : SequenceFlags, PacketSequenceCount
                                -> PacketSC
endtype
```

2.2.2.2.1 Sequence Flags Type

The Sequence Flags may be set by the user of the Packet service to indicate that the User Data contained within the P_SDU is a segment of a larger set of application data; the flags are not part of the Path protocol. Constants are defined for continuation, first, last, and unsegmented values.

```

type      SequenceFlags is Bit, Boolean
sorts     SequenceFlags
opns      SequenceFlags          : Bit, Bit -> SequenceFlags
          Bit1, Bit2             : SequenceFlags -> Bit
          _Eq_                   : SequenceFlags, SequenceFlags -> Bool
          _Ne_                   : SequenceFlags, SequenceFlags -> Bool
          PacketSequenceContinuation
                                : -> SequenceFlags
          PacketSequenceFirstSeg  : -> SequenceFlags
          PacketSequenceLastSeg   : -> SequenceFlags
          PacketSequenceUnSeg     : -> SequenceFlags

eqns      forall b1, b2, b3, b4 : Bit, SF1, SF2 : SequenceFlags

          ofsort SequenceFlags

          PacketSequenceContinuation = SequenceFlags(0,0) ;
          PacketSequenceFirstSeg     = SequenceFlags(0,1) ;
          PacketSequenceLastSeg      = SequenceFlags(1,0) ;
          PacketSequenceUnSeg        = SequenceFlags(1,1) ;

          ofsort Bit

          Bit1(SequenceFlags(b1, b2))          = b1 ;
          Bit2(SequenceFlags(b1, b2))          = b2 ;

          ofsort Bool

          SequenceFlags(b1,b2) Eq SequenceFlags(b3,b4)
            = (b1 Eq b3) And (b2 Eq b4) ;

          SF1 Ne SF2 = Not(SF1 Eq SF2) ;

endtype

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.2.2.2.2 Sequence Count

Sequence Count is a sequential count of each CP_PDU generated on a particular LDP. Initial sequence count and next functions included for stepping through sequence counts are defined.

```
type PacketSequenceCount is Bit, Boolean
sorts PacketSequenceCount
opns PacketSequenceCount : Bit, Bit, Bit, Bit, Bit, Bit,
                             Bit, Bit, Bit, Bit, Bit, Bit,
                             Bit, Bit -> PacketSequenceCount

Bit1, Bit2, Bit3, Bit4,
Bit5, Bit6, Bit7, Bit8,
Bit9, Bit10, Bit11,
Bit12, Bit13, Bit14 : PacketSequenceCount -> Bit
Next : PacketSequenceCount -> PacketSequenceCount
_Eq_, _Ne_ : PacketSequenceCount,
            PacketSequenceCount -> Bool

eqns forall b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11,
        b12, b13, b14 : Bit, SC1, SC2 : PacketSequenceCount

ofsort Bit

Bit1(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b1 ;

Bit2(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b2 ;

Bit3(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b3 ;

Bit4(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b4 ;

Bit5(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b5 ;

Bit6(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b6 ;

Bit7(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b7 ;

Bit8(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b8 ;

Bit9(PacketSequenceCount(b1, b2, b3, b4, b5,
                          b6, b7, b8, b9, b10,
                          b11, b12, b13, b14)) = b9 ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
Bit10(PacketSequenceCount(b1, b2, b3, b4, b5,
                           b6, b7, b8, b9, b10,
                           b11, b12, b13, b14))      = b10 ;

Bit11(PacketSequenceCount(b1, b2, b3, b4, b5,
                           b6, b7, b8, b9, b10,
                           b11, b12, b13, b14))      = b11 ;

Bit12(PacketSequenceCount(b1, b2, b3, b4, b5,
                           b6, b7, b8, b9, b10,
                           b11, b12, b13, b14))      = b12 ;

Bit13(PacketSequenceCount(b1, b2, b3, b4, b5,
                           b6, b7, b8, b9, b10,
                           b11, b12, b13, b14))      = b13 ;

Bit14(PacketSequenceCount(b1, b2, b3, b4, b5,
                           b6, b7, b8, b9, b10,
                           b11, b12, b13, b14))      = b14 ;

ofsort Bool

SC1 Eq SC2      = (Bit1(SC1) Eq Bit1(SC2)) And
                  (Bit2(SC1) Eq Bit2(SC2)) And
                  (Bit3(SC1) Eq Bit3(SC2)) And
                  (Bit4(SC1) Eq Bit4(SC2)) And
                  (Bit5(SC1) Eq Bit5(SC2)) And
                  (Bit6(SC1) Eq Bit6(SC2)) And
                  (Bit7(SC1) Eq Bit7(SC2)) And
                  (Bit8(SC1) Eq Bit8(SC2)) And
                  (Bit9(SC1) Eq Bit9(SC2)) And
                  (Bit10(SC1) Eq Bit10(SC2)) And
                  (Bit11(SC1) Eq Bit11(SC2)) And
                  (Bit12(SC1) Eq Bit12(SC2)) And
                  (Bit13(SC1) Eq Bit13(SC2)) And
                  (Bit14(SC1) Eq Bit14(SC2)) ;

SC1 Ne SC2      = Not(SC1 Eq SC2) ;

ofsort PacketSequenceCount

Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12, b13, 0)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12, b13, 1) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12, 0, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, b12, 1, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, 0, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11, 1, 0, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, 0, 1, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, 1, 0, 0, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, 0, 1, 1, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, b9, 1, 0, 0, 0, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, 0, 1, 1, 1, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, b8, 1, 0, 0, 0, 0, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, 0, 1, 1, 1, 1, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, b7, 1, 0, 0, 0, 0, 0, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, b6, 0, 1, 1, 1, 1, 1, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, b6, 1, 0, 0, 0, 0, 0, 0, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, b5, 0, 1, 1, 1, 1, 1, 1, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, b5, 1, 0, 0, 0, 0, 0, 0, 0, 0) ;
Next(PacketSequenceCount(b1, b2, b3, b4, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1)) =
  PacketSequenceCount(b1, b2, b3, b4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0) ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
Next(PacketSequenceCount(b1,b2,b3,0,1,1,1,1,1,1,1,1,1,1,1)) =  
    PacketSequenceCount(b1,b2,b3,1,0,0,0,0,0,0,0,0,0,0,0) ;  
Next(PacketSequenceCount(b1,b2,0,1,1,1,1,1,1,1,1,1,1,1,1)) =  
    PacketSequenceCount(b1,b2,1,0,0,0,0,0,0,0,0,0,0,0,0) ;  
Next(PacketSequenceCount(b1,0,1,1,1,1,1,1,1,1,1,1,1,1,1)) =  
    PacketSequenceCount(b1,1,0,0,0,0,0,0,0,0,0,0,0,0,0) ;  
Next(PacketSequenceCount(0,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =  
    PacketSequenceCount(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;  
Next(PacketSequenceCount(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =  
    PacketSequenceCount(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;  
endtype
```

2.2.2.3 Packet Length

Packet Length is the number of octets remaining in the CP_PDU. In addition to usual operations, conversions of natural number to packet length and packet length to natural number are defined.

```

type      PacketLength is Bit, Boolean, NaturalNumber
sorts     PacketLength
opns      PacketLength          : Bit, Bit, Bit, Bit, Bit, Bit,
                                Bit, Bit, Bit, Bit, Bit, Bit,
                                Bit, Bit, Bit, Bit -> PacketLength

Bit1, Bit2, Bit3, Bit4,
Bit5, Bit6, Bit7, Bit8,
Bit9, Bit10, Bit11,
Bit12, Bit13, Bit14,
Bit15, Bit16          : PacketLength -> Bit
_Eq_, _Ne_           : PacketLength, PacketLength -> Bool
ConvertNatToPL       : Nat -> PacketLength
ConvertPLToNat       : PacketLength -> Nat
Next                 : PacketLength -> PacketLength

eqns      forall b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11,
              b12, b13, b14, b15, b16 : Bit, PL1, PL2 : PacketLength,
              N : Nat

ofsort Bit

Bit1(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b1 ;

Bit2(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b2 ;

Bit3(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b3 ;

Bit4(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b4 ;

Bit5(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b5 ;

Bit6(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b6 ;

Bit7(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b7 ;

Bit8(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b8 ;

Bit9(PacketLength(b1, b2, b3, b4, b5, b6,
                  b7, b8, b9, b10, b11, b12,
                  b13, b14, b15, b16))      = b9 ;

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
Bit10(PacketLength(b1, b2, b3, b4, b5, b6,  
                   b7, b8, b9, b10, b11, b12,  
                   b13, b14, b15, b16))      = b10 ;  
  
Bit11(PacketLength(b1, b2, b3, b4, b5, b6,  
                   b7, b8, b9, b10, b11, b12,  
                   b13, b14, b15, b16))      = b11 ;  
  
Bit12(PacketLength(b1, b2, b3, b4, b5, b6,  
                   b7, b8, b9, b10, b11, b12,  
                   b13, b14, b15, b16))      = b12 ;  
  
Bit13(PacketLength(b1, b2, b3, b4, b5, b6,  
                   b7, b8, b9, b10, b11, b12,  
                   b13, b14, b15, b16))      = b13 ;  
  
Bit14(PacketLength(b1, b2, b3, b4, b5, b6,  
                   b7, b8, b9, b10, b11, b12,  
                   b13, b14, b15, b16))      = b14 ;  
  
Bit15(PacketLength(b1, b2, b3, b4, b5, b6,  
                   b7, b8, b9, b10, b11, b12,  
                   b13, b14, b15, b16))      = b15 ;  
  
Bit16(PacketLength(b1, b2, b3, b4, b5, b6,  
                   b7, b8, b9, b10, b11, b12,  
                   b13, b14, b15, b16))      = b16 ;
```

ofsort Bool

```
PL1 Eq PL2      = (Bit1(PL1) Eq Bit1(PL2)) And  
                  (Bit2(PL1) Eq Bit2(PL2)) And  
                  (Bit3(PL1) Eq Bit3(PL2)) And  
                  (Bit4(PL1) Eq Bit4(PL2)) And  
                  (Bit5(PL1) Eq Bit5(PL2)) And  
                  (Bit6(PL1) Eq Bit6(PL2)) And  
                  (Bit7(PL1) Eq Bit7(PL2)) And  
                  (Bit8(PL1) Eq Bit8(PL2)) And  
                  (Bit9(PL1) Eq Bit9(PL2)) And  
                  (Bit10(PL1) Eq Bit10(PL2)) And  
                  (Bit11(PL1) Eq Bit11(PL2)) And  
                  (Bit12(PL1) Eq Bit12(PL2)) And  
                  (Bit13(PL1) Eq Bit13(PL2)) And  
                  (Bit14(PL1) Eq Bit14(PL2)) And  
                  (Bit15(PL1) Eq Bit15(PL2)) And  
                  (Bit16(PL1) Eq Bit16(PL2)) ;
```

```
PL1 Ne PL2      = Not(PL1 Eq PL2) ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

ofsort PacketLength

(N Eq 0) =>

ConvertNatToPL(N) = PacketLength(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

(N Ne 0) =>

ConvertNatToPL(N) = Next(ConvertNatToPL(Pred(N)));

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,0)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,1) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,0,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,1,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,0,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,1,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,0,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,1,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,0,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,1,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,0,1,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,1,0,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,0,1,1,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,b9,1,0,0,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,0,1,1,1,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,1,0,0,0,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,b7,0,1,1,1,1,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,b7,1,0,0,0,0,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,b6,0,1,1,1,1,1,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,b6,1,0,0,0,0,0,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,b5,0,1,1,1,1,1,1,1,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,b5,1,0,0,0,0,0,0,0,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,b4,0,1,1,1,1,1,1,1,1,1,1,1,1)) =
PacketLength(b1,b2,b3,b4,1,0,0,0,0,0,0,0,0,0,0,0,0) ;

Next(PacketLength(b1,b2,b3,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
PacketLength(b1,b2,b3,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;

Next(PacketLength(b1,b2,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
PacketLength(b1,b2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;

Next(PacketLength(b1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
PacketLength(b1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;

Next(PacketLength(0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
PacketLength(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;

Next(PacketLength(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
PacketLength(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

ofsort Nat

```
ConvertPLToNat(PacketLength(b1,b2,b3,b4,b5,b6,b7,b8,  
                             b9,b10,b11,b12,b13,b14,b15,b16)) =  
  (((((((((((((((  
    NatNum(b1) * 2) + NatNum(b2) * 2) + NatNum(b3) * 2) +  
    NatNum(b4) * 2) + NatNum(b5) * 2) + NatNum(b6) * 2) +  
    NatNum(b7) * 2) + NatNum(b8) * 2) + NatNum(b9) * 2) +  
    NatNum(b10) * 2) + NatNum(b11) * 2) + NatNum(b12) * 2) +  
    NatNum(b13) * 2) + NatNum(b14) * 2) + NatNum(b15) * 2) +  
    NatNum(b16) ;
```

endtype

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.2.3 User Data

The Secondary Header part of the Version-1 CCSDS Packet is not used in these specifications and is considered a part of the User Data.

The User Data field is of unknown internal form and is an integral number of octets in length; the OctetString ADT (5.7) will be used for this purpose.

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

2.3 PATH SUBNETWORK DATA TYPES

2.3.1 CCSDS Packet Service Data Unit

The Packet Service Data Unit (P_SDU) is a Version-1 CCSDS Packet that has been created by the Path protocol. As such, a separate ADT is not needed, and the CCSDSPacket ADT (2.2.1) will be used instead.

3 VCLC DATA TYPES

3.1 VCLC SERVICE DATA TYPES

3.1.1 Encapsulation Service Data Unit

The Encapsulation Service Data Unit (E_SDU) is a delimited, octet-aligned data unit which, since it is not formatted as a Version-1 CCSDS Packet, has a format and content that are both unknown to the VCLC sublayer. The E_SDU is passed either (1) from the upper layer as a parameter of the service data primitive E_UNITDATA.request for encapsulation or (2) to the upper layer as a parameter of the service data primitive E_UNITDATA.indication after de-encapsulation. The E_SDU may have any format and be of any length which is an integral number of octets, up to 65,536 (2^{16}) maximum. The encapsulation procedure must simply preserve and pass the E_SDU without modification.

As the E_SDU is functionally equivalent to the OctetString ADT (5.7), the OctetString ADT will be used instead.

3.1.2 E_SDU Loss Flag

The E_SDU Loss Flag is a Boolean value used to indicate the loss of E_SDU data during the de-encapsulation process.

```

type      ESDULossFlag is Boolean
sorts     ESDULossFlag
opns      ESDULost           : -> ESDULossFlag
          ESDUNotLost       : -> ESDULossFlag
          _Eq_              : ESDULossFlag,
                           ESDULossFlag -> Bool

eqns

ofsort   Bool

          ESDULost Eq ESDULost           = true           ;
          ESDUNotLost Eq ESDUNotLost     = true           ;
          ESDULost Eq ESDUNotLost       = false          ;
          ESDUNotLost Eq ESDULost       = false          ;

endtype
    
```

3.1.3 Multiplexing Service Data Unit

The Multiplexing Service Data Unit (M_SDU) is formatted as a Version-1 CCSDS Packet. The M_SDU is received from the upper layer as a parameter of the service data primitive.

As the M_SDU is functionally equivalent to the CCSDSPacket ADT, the CCSDSPacket ADT will be used for the M_SDU.

3.1.4 Bitstream Data

The Bitstream Data which accompany a BITSTREAM.request or a BITSTREAM.indication are undelimited strings of bits.

The Bitstream Data field is of unknown internal form; the BitString ADT (5.6) will be used for Bitstream Data.

3.1.5 PCID

The Packet Channel Identifier (PCID) is locally expressed by the APID field in the Version-1 CCSDS Packet Header. As the PCID is identical to the APID, the APID will be used for the PCID.

3.1.6 Bitstream Data Loss Flag

The Bitstream Data Loss Flag is used to indicate the loss of Bitstream data.

```

type BitstreamDataLossFlag is Boolean, VCDULossFlag
sorts BitstreamDataLossFlag
opns  BitstreamDataLost      : -> BitstreamDataLossFlag
      BitstreamDataNotLost   : -> BitstreamDataLossFlag
      ConvertVCDULFtoBitstreamLF : VCDULossFlag
                                       -> BitstreamDataLossFlag
      _Eq_                    : BitstreamDataLossFlag,
                                       BitstreamDataLossFlag -> Bool
eqns
  ofsort Bool
    BitstreamDataLost eq BitstreamDataLost      = true ;
    BitstreamDataNotLost eq BitstreamDataNotLost = true ;
    BitstreamDataLost eq BitstreamDataNotLost   = false ;
    BitstreamDataNotLost eq BitstreamDataLost   = false ;

  ofsort BitstreamDataLossFlag
    ConvertVCDULFtoBitstreamLF(
      VCDULost)                    = BitstreamDataLost;

    ConvertVCDULFtoBitstreamLF(
      VCDUNotLost)                  = BitstreamDataNotLost;

endtype

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

3.2 VCLC PROTOCOL DATA TYPES

3.2.1 Encapsulation Protocol Data Unit

The Encapsulation Protocol Data Unit (E_PDU) is passed to the lower layer as a parameter of the service data primitive.

As the E_PDU is functionally equivalent to the CCSDSPacket ADT, the CCSDSPacket ADT will be used for the E_PDU.

3.2.2 Multiplexing Protocol Data Unit

The Multiplexing Protocol Data Unit (M_PDU) has the following structure:

M_PDU Header	2 octets
Spare	5 bits
First Header Pointer	11 bits
Packet Zone (contains CCSDS packets or portions thereof)	fixed length/VC

The following defines the M_PDU format. The M_PDU contains a M_PDU header field and an M_PDU Packet Zone (CCSDS Packets).

```
type    MPDU is MPDUHeader, OctetString
sorts   MPDU

opns    MakeMPDU          : MPDUHeader, OctetString -> MPDU
        GetMPDUHeader    : MPDU -> MPDUHeader
        GetMPDUPZ        : MPDU -> OctetString
        ConvertMPDUtoOS  : MPDU -> OctetString
        ConvertOStoMPDU : OctetString -> MPDU

eqns    forall MH1 : MPDUHeader,
        MPZ1, OS1 : OctetString

        ofsort MPDUHeader

        GetMPDUHeader(MakeMPDU(MH1, MPZ1)) = MH1 ;

        ofsort OctetString

        GetMPDUPZ(MakeMPDU(MH1, MPZ1)) = MPZ1 ;

        ConvertMPDUtoOS(
            MakeMPDU(MH1, MPZ1))          = Append( MPZ1,
                                                    ConvertMPDUHeadertoOS(MH1));

        ofsort MPDU

        ConvertOStoMPDU(OS1) = MakeMPDU(
                                ConvertOStoMPDUHeader(
                                    First(OS1),
                                    Nth(OS1, 2)
                                ),
                                StripOctets(OS1, 2)
                            )

endtype
```

3.2.2.1 M_PDU Header

The following defines the Multiplexing Protocol Data Unit (M_PDU) header field. The M_PDU header contains the Spares field and the First Header Pointer (FHP) field. The M_PDU header is 16 bits in length.

```

type      MPDUHeader is MPDUSpare, FirstHeaderPointer, Octet, OctetString

sorts     MPDUHeader

opns      MakeMPDUHeader      : MPDUSpare, MPDUFHP -> MPDUHeader
          GetFHP              : MPDUHeader -> MPDUFHP
          ConvertMPDUHeadertoOS : MPDUHeader -> OctetString
          ConvertOSToMPDUHeader : Octet, Octet -> MPDUHeader

eqns      forall FHP1 : MPDUFHP,
          Spare1 : MPDUSpare,
          O1, O2 : Octet

          ofsort MPDUFHP

          GetFHP(MakeMPDUHeader(Spare1, FHP1)) = FHP1 ;

          ofsort OctetString

          ConvertMPDUHeadertoOS(
            MakeMPDUHeader(Spare1, FHP1)) = AddFront(
              Octet(
                0,
                0,
                0,
                0,
                0,
                Bit1(FHP1),
                Bit2(FHP1),
                Bit3(FHP1) ),
              AddFront(
                Octet(
                  Bit4(FHP1),
                  Bit5(FHP1),
                  Bit6(FHP1),
                  Bit7(FHP1),
                  Bit8(FHP1),
                  Bit9(FHP1),
                  Bit10(FHP1),
                  Bit11(FHP1) ),
                NullOS));

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

ofsort MPDUHeader

```
ConvertOSToMPDUHeader(O1,O2) = MakeMPDUHeader(  
    MakeMPDUSpare(  
        Bit1(O1),  
        Bit2(O1),  
        Bit3(O1),  
        Bit4(O1),  
        Bit5(O1)  
    ),  
    MakeMPDUFHP(  
        Bit6(O1),  
        Bit7(O1),  
        Bit8(O1),  
        Bit1(O2),  
        Bit2(O2),  
        Bit3(O2),  
        Bit4(O2),  
        Bit5(O2),  
        Bit6(O2),  
        Bit7(O2),  
        Bit8(O2)  
    )  
);
```

endtype

3.2.2.1.1 M_PDU Spare Field

The following defines the Spare field, which is part of the M_PDU Header field. This field is not currently used by the protocol; it is being held in reserve.

```

type      MPDUSpare is Bit, Boolean

sorts     MPDUSpare

opns      MPDUSpare           : -> MPDUSpare
          MakeMPDUSpare       : Bit, Bit, Bit, Bit, Bit
                               -> MPDUSpare

eqns      forall b1, b2, b3, b4, b5: Bit, MS1, MS2: MPDUSpare
          ofsort MPDUSpare
          MPDUSpare = MakeMPDUSpare(0, 0, 0, 0, 0) ;

endtype
    
```

3.2.2.1.2 M_PDU First Header Pointer

The First Header Pointer (FHP) is a Natural Number (Nat) which is part of the M_PDU Header field. The FHP consists of 11 bits. It is a pointer to the first octet of the first CCSDS Packet Header in the Packet Zone. It is zero when the first octet of the Packet Zone is the first octet of the first CCSDS Packet Header, so it acts as an offset value within the Packet Zone. If the Packet Zone does not contain a CCSDS packet header, but does contain a portion of a CCSDS packet, the FHP is set to 'all ones'. If the Packet Zone contains only fill data, the FHP is set to 'all ones minus one'.

```

type      FirstHeaderPointer is Bit, Boolean, NaturalNumber, BitString
sorts     MPDUFHP
opns      NoPacketHeaderFHP      : -> MPDUFHP
          FillFHP                 : -> MPDUFHP
          NullFHP                  : -> MPDUFHP
          MakeMPDUFHP              : Bit, Bit, Bit, Bit,
                                   Bit, Bit, Bit, Bit,
                                   Bit, Bit, Bit -> MPDUFHP

          Bit1, Bit2, Bit3, Bit4,
          Bit5, Bit6, Bit7, Bit8,
          Bit9, Bit10, Bit11      : MPDUFHP -> Bit
          ConvertFHPtoNat         : MPDUFHP -> Nat
          ConvertNattoFHP        : Nat -> MPDUFHP
          _Eq_, _Ne_              : MPDUFHP, MPDUFHP -> Bool
          NextFHP                 : MPDUFHP -> MPDUFHP

eqns      forall b1, b2, b3, b4, b5, b6, b7,b8, b9, b10, b11 : Bit,
          FHP1, FHP2 : MPDUFHP,
          Nat1 : Nat,
          BS1 : BitString

          ofsort MPDUFHP

          NoPacketHeaderFHP      = MakeMPDUFHP(1,1,1,1,1,1,1,1,1,1,1) ;
          FillFHP                 = MakeMPDUFHP(1,1,1,1,1,1,1,1,1,1,0) ;
          NullFHP                 = MakeMPDUFHP(0,0,0,0,0,0,0,0,0,0,0) ;

          NextFHP(
            MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,0))
            = MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,1) ;

          NextFHP(
            MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,0,1))
            = MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,1,0) ;

          NextFHP(
            MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,0,1,1))
            = MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,1,0,0) ;

          NextFHP(
            MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,0,1,1,1))
            = MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,1,0,0,0) ;

          NextFHP(
            MakeMPDUFHP(b1,b2,b3,b4,b5,b6,0,1,1,1,1))
            = MakeMPDUFHP(b1,b2,b3,b4,b5,b6,1,0,0,0,0) ;

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
NextFHP(
  MakeMPDUFHP(b1,b2,b3,b4,b5,0,1,1,1,1,1))
  = MakeMPDUFHP(b1,b2,b3,b4,b5,1,0,0,0,0,0) ;

NextFHP(
  MakeMPDUFHP(b1,b2,b3,b4,0,1,1,1,1,1,1))
  = MakeMPDUFHP(b1,b2,b3,b4,1,0,0,0,0,0,0) ;

NextFHP(
  MakeMPDUFHP(b1,b2,b3,0,1,1,1,1,1,1,1))
  = MakeMPDUFHP(b1,b2,b3,1,0,0,0,0,0,0,0) ;

NextFHP(
  MakeMPDUFHP(b1,b2,0,1,1,1,1,1,1,1,1))
  = MakeMPDUFHP(b1,b2,1,0,0,0,0,0,0,0,0) ;

NextFHP(
  MakeMPDUFHP(b1,0,1,1,1,1,1,1,1,1,1))
  = MakeMPDUFHP(b1,1,0,0,0,0,0,0,0,0,0) ;

NextFHP(
  MakeMPDUFHP(0,1,1,1,1,1,1,1,1,1,1))
  = MakeMPDUFHP(1,0,0,0,0,0,0,0,0,0,0) ;

ConvertNattoFHP(0) = NullFHP ;
ConvertNattoFHP(succ(Nat1)) = NextFHP(ConvertNattoFHP(Nat1)) ;

ofsort Nat

ConvertFHPtoNat(FHP1) = (((((((((((NatNum(Bit11(FHP1))) +
  (2 * NatNum(Bit10(FHP1)))) +
  ((2 ** 2) * NatNum(Bit9(FHP1)))) +
  ((2 ** 3) * NatNum(Bit8(FHP1)))) +
  ((2 ** 4) * NatNum(Bit7(FHP1)))) +
  ((2 ** 5) * NatNum(Bit6(FHP1)))) +
  ((2 ** 6) * NatNum(Bit5(FHP1)))) +
  ((2 ** 7) * NatNum(Bit4(FHP1)))) +
  ((2 ** 8) * NatNum(Bit3(FHP1)))) +
  ((2 ** (8+1)) * NatNum(Bit2(FHP1)))) +
  ((2 ** (8+2)) * NatNum(Bit1(FHP1)))) ;

ofsort Bit

Bit1(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b1 ;
Bit2(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b2 ;
Bit3(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b3 ;
Bit4(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b4 ;
Bit5(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b5 ;
Bit6(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b6 ;
Bit7(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b7 ;
Bit8(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b8 ;
Bit9(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b9 ;
Bit10(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b10 ;
Bit11(MakeMPDUFHP(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11)) = b11 ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

ofsort Bool

```
FHP1 Eq FHP2 = ((((((((((Bit1(FHP1) Eq Bit1(FHP2))
and
(Bit2(FHP1) Eq Bit2(FHP2)))
and
(Bit3(FHP1) Eq Bit3(FHP2)))
and
(Bit4(FHP1) Eq Bit4(FHP2)))
and
(Bit5(FHP1) Eq Bit5(FHP2)))
and
(Bit6(FHP1) Eq Bit6(FHP2)))
and
(Bit7(FHP1) Eq Bit7(FHP2)))
and
(Bit8(FHP1) Eq Bit8(FHP2)))
and
(Bit9(FHP1) Eq Bit9(FHP2)))
and
(Bit10(FHP1) Eq Bit10(FHP2)))
and
(Bit11(FHP1) Eq Bit11(FHP2))))))));
```

```
FHP1 Ne FHP2 = not(FHP1 Eq FHP2)      ;
```

endtype

3.2.2.2 M_PDU Packet Zone

The M_PDU Packet Zone is an octet-delimited string of octets. Thus, the OctetString ADT (5.7) will be used for the M_PDU Packet Zone.

3.2.3 Bitstream Protocol Data Unit

The following defines the Bitstream Protocol Data Unit (B_PDU) format. The B_PDU contains a B_PDU Header field and strings of bits (BitString data).

```

type      BPDU is BPDUHeader, BitString, OctetString, BitFillData,
                NaturalNumber
sorts     BPDU
opns      MakeBPDU          : BPDUHeader, BitString
                -> BPDU
                MakeFillBPDU      : Nat, BitString -> BPDU
                GetBPDUHeader     : BPDU -> BPDUHeader
                GetBDZ            : BPDU -> BitString
                ConvertBPDUtoOS   : BPDU -> OctetString
                ConvertOStoBPDU   : OctetString -> BPDU
eqns      forall BH1 : BPDUHeader,
                BDZ1, FillPattern : BitString,
                FillLength : Nat,
                OS1 : OctetString

ofsort BPDUHeader
GetBPDUHeader(MakeBPDU(BH1, BDZ1)) = BH1 ;

ofsort BitString
GetBDZ(MakeBPDU(BH1, BDZ1)) = BDZ1 ;

ofsort OctetString
ConvertBPDUtoOS(MakeBPDU(BH1, BDZ1)) = Append(
                ConvertBStoOS(
                    BDZ1),
                ConvertBHtoOS(
                    BH1)
                );

ofsort BPDU
ConvertOStoBPDU(OS1) = MakeBPDU(
                ConvertOStoBH(
                    AddFront(Nth(OS1,1),
                    AddFront(Nth(OS1,2),
                    NullOS))),
                ConvertOStoBS(
                    StripOctets(OS1,2)
                )
                );

                MakeFillBPDU(FillLength,
                FillPattern) = MakeBPDU(MakeBPDUHeader(BPDUSpare,
                OnlyFillDataBDP),
                MakeBitFillData(FillPattern,
                FillLength)) ;

endtype

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

3.2.3.1 B_PDU Header

The following is a definition of the Bitstring Protocol Data Unit (B_PDU) Header field. The B_PDU Header contains the Spares field and the Bitstring Data Pointer (BDP) field. It is anticipated that the B_PDU Header will be a total of 16 bits in length.

```
type      BPDUHeader is BPDUSpare, BitstreamDataPointer, OctetString
sorts     BPDUHeader

opns      MakeBPDUHeader          : BPDUSpare, BitstreamDataPointer
                                         -> BPDUHeader
          GetBitstreamDataPointer : BPDUHeader
                                         -> BitstreamDataPointer
          ConvertBHtoOS           : BPDUHeader -> OctetString
          ConvertOSToBH           : OctetString -> BPDUHeader

eqns      forall BDP1: BitstreamDataPointer,
            Spare1: BPDUSpare,
            OS1 : OctetString

ofsort BitstreamDataPointer

GetBitstreamDataPointer(MakeBPDUHeader(Spare1, BDP1)) = BDP1 ;

ofsort OctetString

ConvertBHtoOS(
  MakeBPDUHeader(Spare1, BDP1)) = AddFront(
  Octet(
    0,
    0,
    Bit1(BDP1),
    Bit2(BDP1),
    Bit3(BDP1),
    Bit4(BDP1),
    Bit5(BDP1),
    Bit6(BDP1)
  ),
  AddFront(
    Octet(
      Bit7(BDP1),
      Bit8(BDP1),
      Bit9(BDP1),
      Bit10(BDP1),
      Bit11(BDP1),
      Bit12(BDP1),
      Bit13(BDP1),
      Bit14(BDP1)
    ),
    NullOS
  )
) ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

ofsort BPDUHeader

```
ConvertOSToBH(OS1)          = MakeBPDUHeader(  
    MakeBPDUSpare(  
        Bit1(First(OS1)),  
        Bit2(First(OS1))  
    ),  
    MakeBitstreamDataPointer(  
        Bit3(First(OS1)),  
        Bit4(First(OS1)),  
        Bit5(First(OS1)),  
        Bit6(First(OS1)),  
        Bit7(First(OS1)),  
        Bit8(First(OS1)),  
        Bit1(Nth(OS1,2)),  
        Bit2(Nth(OS1,2)),  
        Bit3(Nth(OS1,2)),  
        Bit4(Nth(OS1,2)),  
        Bit5(Nth(OS1,2)),  
        Bit6(Nth(OS1,2)),  
        Bit7(Nth(OS1,2)),  
        Bit8(Nth(OS1,2))  
    )  
);
```

endtype

3.2.3.1.1 B_PDU Spare Field

The B_PDU Spare field in the Header is not used currently. It is 2 bits long.

```
type BPDUSpare is Bit, Boolean

sorts BPDUSpare

opns  BPDUSpare           : -> BPDUSpare
      MakeBPDUSpare      : Bit, Bit -> BPDUSpare

eqns  forall b1, b2: Bit, BS1, BS2: BPDUSpare
      ofsort BPDUSpare
      BPDUSpare = MakeBPDUSpare(0, 0) ;

endtype
```


3.2.3.1.2 Bitstream Data Pointer

The Bitstream Data Pointer (BDP) is part of the B_PDU Header field. The BDP consists of 14 bits. It is a count of the number of user-supplied bits contained in the Bitstream Data Zone (BDZ), except when the BDZ contains only fill data or no fill data. In the case of a fill-only BDZ, the BDP is set to 'all ones minus one'; in the case of a no-fill BDZ, the BDP is set to 'all ones'.

```

type      BitstreamDataPointer is Bit, Boolean

sorts     BitstreamDataPointer

opns      OnlyFillDataBDP           : -> BitstreamDataPointer
          NoFillDataBDP            : -> BitstreamDataPointer
          NullBDP                  : -> BitstreamDataPointer
          MakeBitstreamDataPointer : Bit, Bit, Bit, Bit,
                                     Bit, Bit, Bit, Bit,
                                     Bit, Bit, Bit, Bit,
                                     Bit, Bit -> BitstreamDataPointer

          Bit1, Bit2, Bit3, Bit4,
          Bit5, Bit6, Bit7, Bit8,
          Bit9, Bit10, Bit11,
          Bit12, Bit13, Bit14       : BitstreamDataPointer -> Bit
          _Eq_, _Ne_                : BitstreamDataPointer,
                                     BitstreamDataPointer -> Bool
          Next, Pred                 : BitstreamDataPointer ->
                                     BitstreamDataPointer
          ConvertNattoBDP            : Nat -> BitstreamDataPointer
          ConvertBDPToNat           : BitStreamDataPointer -> Nat

eqns      forall b1, b2, b3, b4, b5, b6, b7,
          b8, b9, b10, b11, b12, b13, b14 : Bit,
          BDP1, BDP2 : BitstreamDataPointer,
          Nat1 : Nat

          ofsort Bit

          Bit1(MakeBitstreamDataPointer(
            b1,b2,b3,b4,b5,b6,b7,b8,
            b9,b10,b11,b12,b13,b14))      = b1                ;

          Bit2(MakeBitstreamDataPointer(
            b1,b2,b3,b4,b5,b6,b7,b8,
            b9,b10,b11,b12,b13,b14))      = b2                ;

          Bit3(MakeBitstreamDataPointer(
            b1,b2,b3,b4,b5,b6,b7,b8,
            b9,b10,b11,b12,b13,b14))      = b3                ;

          Bit4(MakeBitstreamDataPointer(
            b1,b2,b3,b4,b5,b6,b7,b8,
            b9,b10,b11,b12,b13,b14))      = b4                ;

          Bit5(MakeBitstreamDataPointer(
            b1,b2,b3,b4,b5,b6,b7,b8,
            b9,b10,b11,b12,b13,b14))      = b5                ;

          Bit6(MakeBitstreamDataPointer(
            b1,b2,b3,b4,b5,b6,b7,b8,
            b9,b10,b11,b12,b13,b14))      = b6                ;

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
Bit7(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b7      ;

Bit8(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b8      ;

Bit9(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b9      ;

Bit10(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b10     ;

Bit11(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b11     ;

Bit12(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b12     ;

Bit13(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b13     ;

Bit14(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,
  b9,b10,b11,b12,b13,b14))      = b14     ;

ofsort BitstreamDataPointer

OnlyFillDataBDP = MakeBitstreamDataPointer(
  1,1,1,1,1,1,1,1,1,1,1,1,1,0) ;

NullBDP = MakeBitstreamDataPointer(0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;

NoFillDataBDP = MakeBitstreamDataPointer(1,1,1,1,1,1,1,1,1,1,1,1,1,1) ;

Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,0)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,1) ;
Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,0,1)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,1,0) ;
Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,0,1,1)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,1,0,0) ;
Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,0,1,1,1)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,1,0,0,0) ;
Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,0,1,1,1,1)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,1,0,0,0,0) ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,0,1,1,1,1,1)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,1,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,0,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,1,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,0,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(b1,b2,b3,b4,b5,b6,1,0,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(b1,b2,b3,b4,b5,0,1,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(b1,b2,b3,b4,b5,1,0,0,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(b1,b2,b3,b4,0,1,1,1,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(b1,b2,b3,b4,1,0,0,0,0,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(b1,b2,b3,0,1,1,1,1,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(b1,b2,b3,1,0,0,0,0,0,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(b1,b2,0,1,1,1,1,1,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(b1,b2,1,0,0,0,0,0,0,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(b1,0,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(b1,1,0,0,0,0,0,0,0,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(0,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;
Next(MakeBitstreamDataPointer(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)) =
  MakeBitstreamDataPointer(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) ;

Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,1)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,0) ;
Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,1,0)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,0,1) ;
Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,1,0,0)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,0,1,1) ;
Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,1,0,0,0)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,0,1,1,1) ;
Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,b9,1,0,0,0,0)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,b9,0,1,1,1,1) ;
Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,b8,1,0,0,0,0,0)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,b8,0,1,1,1,1,1) ;
Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,b7,1,0,0,0,0,0,0,0)) =
  MakeBitstreamDataPointer(
    b1,b2,b3,b4,b5,b6,b7,0,1,1,1,1,1,1,1) ;
Pred(MakeBitstreamDataPointer(
  b1,b2,b3,b4,b5,b6,1,0,0,0,0,0,0,0,0)) =
  MakeBitstreamDataPointer(b1,b2,b3,b4,b5,b6,0,1,1,1,1,1,1,1,1) ;
Pred(MakeBitstreamDataPointer(b1,b2,b3,b4,b5,1,0,0,0,0,0,0,0,0,0)) =
  MakeBitstreamDataPointer(b1,b2,b3,b4,b5,0,1,1,1,1,1,1,1,1,1) ;
Pred(MakeBitstreamDataPointer(b1,b2,b3,b4,1,0,0,0,0,0,0,0,0,0,0)) =
  MakeBitstreamDataPointer(b1,b2,b3,b4,0,1,1,1,1,1,1,1,1,1,1) ;
```


CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
ConvertNatToBDP(0) = MakeBitstreamDataPointer
                    (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

ConvertNatToBDP(Succ(Nat1)) = Next(ConvertNatToBDP(Nat1)) ;

ofsort Nat

  BDP1 Eq NullBDP =>
ConvertBDPToNat(BDP1)      = 0;

  BDP1 Ne NullBDP =>
ConvertBDPToNat(BDP1)      = Succ(ConvertBDPToNat(Pred(BDP1))) ;

ofsort Bool

BDP1 Eq BDP2              = (Bit1(BDP1) Eq Bit1(BDP2))
                           and
                           (Bit2(BDP1) Eq Bit2(BDP2))
                           and
                           (Bit3(BDP1) Eq Bit3(BDP2))
                           and
                           (Bit4(BDP1) Eq Bit4(BDP2))
                           and
                           (Bit5(BDP1) Eq Bit5(BDP2))
                           and
                           (Bit6(BDP1) Eq Bit6(BDP2))
                           and
                           (Bit7(BDP1) Eq Bit7(BDP2))
                           and
                           (Bit8(BDP1) Eq Bit8(BDP2))
                           and
                           (Bit9(BDP1) Eq Bit9(BDP2))
                           and
                           (Bit10(BDP1) Eq Bit10(BDP2))
                           and
                           (Bit11(BDP1) Eq Bit11(BDP2))
                           and
                           (Bit12(BDP1) Eq Bit12(BDP2))
                           and
                           (Bit13(BDP1) Eq Bit13(BDP2))
                           and
                           (Bit14(BDP1) Eq Bit14(BDP2));

BDP1 Ne BDP2              = not(BDP1 Eq BDP2)      ;

endtype
```

3.2.3.2 B_PDU Bitstream Data Zone

The BitString ADT (5.6) is used to represent the B_PDU Bitstream Data Zone.

4 VCA DATA TYPES

4.1 VCA SERVICE DATA TYPES

4.1.1 VCA_SDU

The VCA_SDU, which may be an M_PDU, a B_PDU, or a SLAP_PDU, is represented using the OctetString ADT (5.7), into which the VCLC converts M_PDUs and B_PDUs before sending to the VCA.

4.1.2 Insert Service Data Unit

The Insert Service Data Unit (IN_SDU) is the service data unit passed to and from users of the Insert service on all VCs. An IN_SDU is an isochronous, octet-aligned data unit of fixed length. Its length at the request (source) interface is always equal to its length at the indication (destination) interface. The IN_SDU is represented using the OctetString ADT.

```

type      InsertLossFlag is Boolean
sorts     InsertLossFlag
opns      INSDULost           : -> InsertLossFlag
          INSDUNotLost       : -> InsertLossFlag
          _Eq_                : InsertLossFlag,
                              InsertLossFlag -> Bool

eqns      forall DLI1, DLI2 : InsertLossFlag

          ofsort Bool

          INSDULost Eq INSDULost           = true           ;
          INSDUNotLost Eq INSDUNotLost     = true           ;
          INSDULost Eq INSDUNotLost       = false          ;
          INSDUNotLost Eq INSDULost       = false          ;

endtype

type      VCDULossFlag is Boolean
sorts     VCDULossFlag
opns      VCDULost           : -> VCDULossFlag
          VCDUNotLost       : -> VCDULossFlag
          _Eq_                : VCDULossFlag,
                              VCDULossFlag -> Bool

eqns      forall DLI1, DLI2 : VCDULossFlag

          ofsort Bool

          VCDULost Eq VCDULost           = true           ;
          VCDUNotLost Eq VCDUNotLost     = true           ;
          VCDULost Eq VCDUNotLost       = false          ;
          VCDUNotLost Eq VCDULost       = false          ;

endtype

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

4.2 VCA PROTOCOL DATA TYPES

The protocol data unit of the Virtual Channel Procedures is the VC_PDU, which is implemented using the CCSDS Virtual Channel Data Unit (VCDU) data structure. A VCDU is composed of a VCDU Primary Header, an optional VCDU Insert Zone, a VCDU Data Unit Zone, and an optional VCDU Trailer.

```
type      VCDU is VCDUPrimaryHeader, OctetString, VCDUTrailer, FillData, Boolean
sorts     VCDU
opns      MakeVCDU                : VCDUPrimaryHeader,
                                   OctetString,
                                   OctetString,
                                   VCDUTrailer,
                                   OctetString -> VCDU
                                   MakeFillVCDU      : OctetString, Nat -> VCDU
                                   NullVCDU         : -> VCDU
                                   GetVCDUPrimaryHeader : VCDU -> VCDUPrimaryHeader
                                   GetInsertZone      : VCDU -> OctetString
                                   SetInsertZone      : OctetString, VCDU -> VCDU
                                   GetVCASDU         : VCDU -> OctetString
                                   GetVCDUTrailer     : VCDU -> VCDUTrailer
                                   SetVCDUTrailer     : VCDUTrailer, VCDU -> VCDU
                                   SetReedSolomon     : OctetString, VCDU -> VCDU
                                   GetReedSolomon     : VCDU -> OctetString
                                   ConvertVCDUtoOS    : VCDU -> OctetString
                                   _Eq_, _Ne_         : VCDU, VCDU -> Bool
eqns      forall VCDU1, VCDU2 : VCDU,
           PH1 : VCDUPrimaryHeader,
           IZ1, IZ2 : OctetString,
           DUZ1 : OctetString,
           RS1, RS2 : OctetString,
           T1, T2 : VCDUTrailer,
           FillData : OctetString,
           FillLength : Nat
ofsort VCDUPrimaryHeader
GetVCDUPrimaryHeader(MakeVCDU(PH1, IZ1, DUZ1, T1, RS1)) = PH1 ;
ofsort VCDU
SetInsertZone(IZ2, MakeVCDU(PH1, IZ1, DUZ1, T1, RS1)) =
  MakeVCDU(PH1, IZ2, DUZ1, T1, RS1) ;
SetVCDUTrailer(T2, MakeVCDU(PH1, IZ1, DUZ1, T1, RS1)) =
  MakeVCDU(PH1, IZ1, DUZ1, T2, RS1) ;
SetReedSolomon(RS2, MakeVCDU(PH1, IZ1, DUZ1, T1, RS1)) =
  MakeVCDU(PH1, IZ1, DUZ1, T1, RS2) ;
MakeFillVCDU(FillData, FillLength)
= MakeVCDU(MakeVCDUPrimaryHeader
  (Version2,
   MakeVCDUID(MakeSCID(0,0,0,0,0,0,0,0),
    FillVCID),
   MakeVCDUCounter(Octet(0,0,0,0,0,0,0,0),
    Octet(0,0,0,0,0,0,0,0),
    Octet(0,0,0,0,0,0,0,0)),
   MakeSignallingField(RealTimeVCDU, VCDUSpare),
   NullOS),
  NullOS,
  MakeFillData(FillData, FillLength),
  NullTrailer,
  NullOS) ;
```


CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
ofsort OctetString

GetInsertZone(MakeVCDU(PH1, IZ1, DUZ1, T1, RS1))      = IZ1 ;
GetVCASDU(MakeVCDU(PH1, IZ1, DUZ1, T1, RS1))        = DUZ1 ;
GetReedSolomon(MakeVCDU(PH1, IZ1, DUZ1, T1, RS1))    = RS1 ;
ConvertVCDUtoOS(MakeVCDU(PH1, IZ1, DUZ1, T1, RS1))   =
  Append(RS1, Append(ConvertVCDUTrailertoOS(T1), Append(DUZ1,
    Append(IZ1, ConvertVCDUHeadertoOS(PH1)))))) ;

ofsort VCDUTrailer

GetVCDUTrailer(MakeVCDU(PH1, IZ1, DUZ1, T1, RS1))    = T1 ;

ofsort Bool

NullVCDU Eq NullVCDU                                = True ;
NullVCDU Eq MakeVCDU(PH1, IZ1, DUZ1, T1, RS1) = False ;
MakeVCDU(PH1, IZ1, DUZ1, T1, RS1) Eq NullVCDU = False ;

VCDU1 Ne VCDU2                                       = Not(VCDU1 Eq VCDU2) ;

endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

4.2.1 VCDU Primary Header

The VCDU Primary Header contains the following fields:

<u>Field:</u>	<u>Length (bits):</u>
VERSION NUMBER	2
VCDU IDENTIFIER:	14
Spacecraft ID (8)	
Virtual Channel ID (6)	
VIRTUAL CHANNEL DATA UNIT COUNTER	24
SIGNALLING FIELD	8
Replay Flag (1)	
Reserved Spares (7)	
VCDU HEADER ERROR CONTROL (optional):	(16)

The total length of the VCDU Primary Header is 48 or 64 bits, depending on whether the optional VCDU Header Error Control field is present.

```
type    VCDUPrimaryHeader is VersionNumber, VCDUID,
        VCDUCounter, SignallingField,
        OctetString
sorts   VCDUPrimaryHeader
opns    MakeVCDUPrimaryHeader : VersionNumber, VCDUID, VCDUCounter,
        SignallingField, OctetString
        -> VCDUPrimaryHeader
        GetVersionNumber      : VCDUPrimaryHeader -> VersionNumber
        GetVCDUID             : VCDUPrimaryHeader -> VCDUID
        GetVCDUCounter        : VCDUPrimaryHeader -> VCDUCounter
        GetSignallingField    : VCDUPrimaryHeader -> SignallingField
        GetVCDUHEC            : VCDUPrimaryHeader -> OctetString
        SetVCDUHEC            : OctetString,
        VCDUPrimaryHeader -> VCDUPrimaryHeader
eqns    ConvertVCDUHeaderToOS : VCDUPrimaryHeader -> OctetString
forall VN1 : VersionNumber,
        VCDUID1 : VCDUID,
        VC1 : VCDUCounter,
        SF1 : SignallingField,
        HEC1, HEC2 : OctetString,
        PH1 : VCDUPrimaryHeader

ofsort VersionNumber

GetVersionNumber(MakeVCDUPrimaryHeader
                 (VN1,VCDUID1,VC1,SF1,HEC1)) = VN1 ;

ofsort VCDUID

GetVCDUID(MakeVCDUPrimaryHeader
          (VN1,VCDUID1,VC1,SF1,HEC1))      = VCDUID1 ;

ofsort VCDUCounter

GetVCDUCounter(MakeVCDUPrimaryHeader
               (VN1,VCDUID1,VC1,SF1,HEC1)) = VC1 ;

ofsort SignallingField

GetSignallingField(MakeVCDUPrimaryHeader
                   (VN1,VCDUID1,VC1,SF1,HEC1)) = SF1 ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
ofsort OctetString

GetVCDUHEC(MakeVCDUPrimaryHeader (VN1,VCDUID1,VC1,SF1,HEC1)) = HEC1 ;

ConvertVCDUHeaderToOS(MakeVCDUPrimaryHeader(VN1,VCDUID1,VC1,SF1,HEC1))
= Append(HEC1,
  AddFront(Octet(Bit1(VN1),
    Bit2(VN1),
    Bit1(GetSCID(VCDUID1)),
    Bit2(GetSCID(VCDUID1)),
    Bit3(GetSCID(VCDUID1)),
    Bit4(GetSCID(VCDUID1)),
    Bit5(GetSCID(VCDUID1)),
    Bit6(GetSCID(VCDUID1))),
  AddFront(Octet(Bit7(GetSCID(VCDUID1)),
    Bit8(GetSCID(VCDUID1)),
    Bit1(GetVCID(VCDUID1)),
    Bit2(GetVCID(VCDUID1)),
    Bit3(GetVCID(VCDUID1)),
    Bit4(GetVCID(VCDUID1)),
    Bit5(GetVCID(VCDUID1)),
    Bit6(GetVCID(VCDUID1))),
  AddFront(Octet1(VC1),
  AddFront(Octet2(VC1),
  AddFront(Octet3(VC1),
  AddFront(Octet(Bit1(GetReplayFlag(SF1)),
    0,
    0,
    0,
    0,
    0,
    0,
    0),
  NullOS)))))) ;

ofsort VCDUPrimaryHeader

SetVCDUHEC(HEC2, MakeVCDUPrimaryHeader(VN1, VCDUID1, VC1, SF1, HEC1)) =
  MakeVCDUPrimaryHeader(VN1, VCDUID1, VC1, SF1, HEC2) ;

endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

4.2.1.1 Version Number

The two Version Number bits (which occupy the two most significant bits of the VCDU Primary Header) are reserved for identification of the VCDU structure.

```
type      VersionNumber is Bit, Boolean
sorts    VersionNumber
opns     Version1      : -> VersionNumber
         Version2      : -> VersionNumber
         MakeVersionNumber : Bit, Bit -> VersionNumber
         Bit1, Bit2      : VersionNumber -> Bit
         _eq_, _ne_      : VersionNumber, VersionNumber -> Bool
eqns     forall b1, b2 : Bit, VN1, VN2 : VersionNumber

         ofsort VersionNumber

         Version1      = MakeVersionNumber(0, 0);
         Version2      = MakeVersionNumber(0, 1);

         ofsort Bit

         Bit1(MakeVersionNumber(b1,b2)) = b1 ;
         Bit2(MakeVersionNumber(b1,b2)) = b2 ;

         ofsort Bool

         VN1 eq VN2      = (Bit1(VN1) eq Bit1(VN2))
                           and
                           (Bit2(VN1) eq Bit2(VN2));
         VN1 ne VN2      = not(VN1 eq VN2) ;

endtype
```

4.2.1.2 VCDU Identifier

The purpose of the VCDU Identifier (VCDU-ID) is to identify the operational spacecraft with which the VCDU is associated, and to identify the Virtual Channel in use. It is composed of the Spacecraft Identifier (SCID) and the Virtual Channel Identifier (VCID).

```

type VCDUID is Boolean, VCID, SCID
sorts VCDUID
opns  MakeVCDUID      : SCID, VCID -> VCDUID
      GetSCID        : VCDUID -> SCID
      GetVCID        : VCDUID -> VCID
      _Eq_           : VCDUID, VCDUID -> Bool
      _Ne_           : VCDUID, VCDUID -> Bool
eqns  forall x, y : VCDUID, a : SCID, b :VCID

      ofsort SCID

      GetSCID(MakeVCDUID(a, b)) = a ;

      ofsort VCID

      GetVCID(MakeVCDUID(a, b)) = b ;

      ofsort Bool

      x Eq y = (GetSCID(x) Eq GetSCID(y))
              And
              (GetVCID(x) Eq GetVCID(y)) ;

      x Ne y = Not(x Eq y) ;
endtype

```

4.2.1.2.1 Spacecraft Identifier

The Spacecraft Identifier (SCID) identifies the various logical entities that provide data to (or receive data from) the VCA sublayer. It also provides the naming domain for the Virtual Channels.

```

type SCID is Bit, Boolean
sorts SCID
opns MakeSCID      : Bit, Bit, Bit, Bit, Bit, Bit, Bit, Bit -> SCID
     _Eq_          : SCID, SCID -> Bool
     _Ne_          : SCID, SCID -> Bool
     Bit1          : SCID -> Bit
     Bit2          : SCID -> Bit
     Bit3          : SCID -> Bit
     Bit4          : SCID -> Bit
     Bit5          : SCID -> Bit
     Bit6          : SCID -> Bit
     Bit7          : SCID -> Bit
     Bit8          : SCID -> Bit
eqns forall x, y : SCID, b1, b2, b3, b4, b5, b6, b7, b8 : Bit

ofsort Bit

Bit1(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b1 ;
Bit2(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b2 ;
Bit3(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b3 ;
Bit4(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b4 ;
Bit5(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b5 ;
Bit6(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b6 ;
Bit7(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b7 ;
Bit8(MakeSCID(b1,b2,b3,b4,b5,b6,b7,b8)) = b8 ;

ofsort Bool

x Eq y      = (Bit1(x) Eq Bit1(y))
             And
             (Bit2(x) Eq Bit2(y))
             And
             (Bit3(x) Eq Bit3(y))
             And
             (Bit4(x) Eq Bit4(y))
             And
             (Bit5(x) Eq Bit5(y))
             And
             (Bit6(x) Eq Bit6(y))
             And
             (Bit7(x) Eq Bit7(y))
             And
             (Bit8(x) Eq Bit8(y)) ;

x Ne y      = Not(x Eq y) ;
endtype

```

4.2.1.2.2 Virtual Channel Identifier

The six-bit Virtual Channel Identifier (VCID) field enables up to 64 Virtual Channels (VCs) to be run concurrently in association with each SCID that is authorized in a particular PCA_PDU.

```

type VCID is Bit, Boolean
sorts VCID
opns  MakeVCID      : Bit, Bit, Bit, Bit, Bit, Bit -> VCID
      FillVCID      : -> VCID
      _Eq_          : VCID, VCID -> Bool
      _Ne_         : VCID, VCID -> Bool
      Bit1         : VCID -> Bit
      Bit2         : VCID -> Bit
      Bit3         : VCID -> Bit
      Bit4         : VCID -> Bit
      Bit5         : VCID -> Bit
      Bit6         : VCID -> Bit
eqns  forall x, y : VCID, b1, b2, b3, b4, b5, b6 : Bit

      ofsort VCID

      FillVCID      = MakeVCID(1,1,1,1,1,1) ;

      ofsort Bit

      Bit1(MakeVCID(b1,b2,b3,b4,b5,b6)) = b1 ;
      Bit2(MakeVCID(b1,b2,b3,b4,b5,b6)) = b2 ;
      Bit3(MakeVCID(b1,b2,b3,b4,b5,b6)) = b3 ;
      Bit4(MakeVCID(b1,b2,b3,b4,b5,b6)) = b4 ;
      Bit5(MakeVCID(b1,b2,b3,b4,b5,b6)) = b5 ;
      Bit6(MakeVCID(b1,b2,b3,b4,b5,b6)) = b6 ;

      ofsort Bool

      x Eq y      = (Bit1(x) Eq Bit1(y))
                    And
                    (Bit2(x) Eq Bit2(y))
                    And
                    (Bit3(x) Eq Bit3(y))
                    And
                    (Bit4(x) Eq Bit4(y))
                    And
                    (Bit5(x) Eq Bit5(y))
                    And
                    (Bit6(x) Eq Bit6(y)) ;

      x Ne y      = Not(x Eq y) ;
endtype

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

4.2.1.3 VCDU Counter Field

The VCDU Counter field provides individual accountability for each of the sixty-four Virtual Channels. The 24-bit field represents a sequential count (modulo 16,777,216) of the total number of VCDUs which have been transmitted on each of the VCs.

```
type      VCDUCounter is CounterOctet, OctetString, Boolean
sorts     VCDUCounter
opns      MakeVCDUCounter      : Octet, Octet, Octet
                                   -> VCDUCounter

      Octet1, Octet2, Octet3    : VCDUCounter -> Octet
      Next                      : VCDUCounter -> VCDUCounter
      _eq_, _ne_                : VCDUCounter,
                                   VCDUCounter -> Bool

eqns      forall O1, O2, O3 : Octet,
              VC1, VC2 : VCDUCounter

ofsort Octet

Octet1(
  MakeVCDUCounter(O1,O2,O3)) = O1          ;
Octet2(
  MakeVCDUCounter(O1,O2,O3)) = O2          ;
Octet3(
  MakeVCDUCounter(O1,O2,O3)) = O3          ;

ofsort VCDUCounter

      Octet3(MakeVCDUCounter(O1, O2, O3)) Ne Octet(1,1,1,1,1,1,1,1) =>
Next(MakeVCDUCounter(O1,O2,O3)) = MakeVCDUCounter(O1,O2,Next(O3)) ;

      (Octet3(MakeVCDUCounter(O1,O2,O3)) Eq Octet(1,1,1,1,1,1,1,1))
And
      (Octet2(MakeVCDUCounter(O1,O2,O3)) Ne Octet(1,1,1,1,1,1,1,1)) =>
Next(MakeVCDUCounter(O1,O2,O3)) = MakeVCDUCounter(O1,Next(O2),Next(O3)) ;

      (Octet3(MakeVCDUCounter(O1,O2,O3)) Eq Octet(1,1,1,1,1,1,1,1))
And
      (Octet2(MakeVCDUCounter(O1,O2,O3)) Eq Octet(1,1,1,1,1,1,1,1)) =>
Next(MakeVCDUCounter(O1,O2,O3)) = MakeVCDUCounter(Next(O1),Next(O2),Next(O3)) ;

ofsort Bool

VC1 eq VC2      = (Octet1(VC1) eq Octet1(VC2))
                  and
                  (Octet2(VC1) eq Octet2(VC2))
                  and
                  (Octet3(VC1) eq Octet3(VC2));
VC1 ne VC2      = not(VC1 eq VC2)          ;

endtype
```


4.2.1.4 Signalling Field

The Signalling field is used to alert the receiver of the VCDU with respect to functions that: (a) may change more rapidly than can be handled by management, or (b) provide a significant cross check against manual or automated setups, for use in VCA sublayer fault detection and isolation. The Signalling field is composed of the Replay Flag and the Reserved Spares field.

```
type      SignallingField is ReplayFlag, VCDUSpare
sorts     SignallingField
opns      MakeSignallingField      : ReplayFlag, VCDUSpare -> SignallingField
          GetReplayFlag            : SignallingField -> ReplayFlag

eqns      forall RF1 : ReplayFlag, Spare1 : VCDUSpare

          ofsort ReplayFlag

          GetReplayFlag(MakeSignallingField(RF1,Spare1)) = RF1 ;

endtype
```

4.2.1.4.1 Replay Flag

The Replay Flag alerts the receiver of the VCDU with respect to its 'realtime' or 'replay' status. Its main purpose is to discriminate between realtime and replay VCDUs transmitted on a particular Physical Channel when they both may use the same VCID.

```

type      ReplayFlag is Bit, Boolean
sorts     ReplayFlag
opns      MakeReplayFlag      : Bit -> ReplayFlag
          RealTimeVCDU        : -> ReplayFlag
          ReplayVCDU          : -> ReplayFlag
          Bit1                 : ReplayFlag -> Bit
          _eq_, _ne_           : ReplayFlag, ReplayFlag -> Bool

eqns      forall b1 : Bit, RF1, RF2 : ReplayFlag

          ofsort ReplayFlag

          RealTimeVCDU = MakeReplayFlag(0) ;
          ReplayVCDU   = MakeReplayFlag(1) ;

          ofsort Bit

          Bit1(MakeReplayFlag(b1))      = b1 ;

          ofsort Bool

          RF1 eq RF2                      = Bit1(RF1) eq Bit1(RF2);
          RF1 ne RF2                      = not(RF1 eq RF2) ;

endtype

```

4.2.1.4.2 Reserved Spares Field

The seven-bit Reserved Spares field is reserved by CCSDS for potential future signalling applications.

```
type VCDUSpare is Bit
sorts VCDUSpare
opns VCDUSpare
    MakeVCDUSpare
    : -> VCDUSpare
    : Bit, Bit, Bit, Bit,
    Bit, Bit, Bit -> VCDUSpare
eqns forall b1, b2, b3, b4, b5, b6, b7 : Bit
    ofsort VCDUSpare
    VCDUSpare = MakeVCDUSpare(0, 0, 0, 0, 0, 0, 0) ;
endtype
```

4.2.1.5 VCDU Header Error Control Field

The VCDU Header Error Control field contains the check symbols of an error detecting and error correcting code used to protect the Version Number field, the VCDU Identifier field, and the Signalling field.

Note – Actual coding is not represented here; zeros are used instead.

```
type VCDUHeaderErrorControl is OctetString
opns VCDUHEC : -> OctetString
     NoVCDUHEC : -> OctetString
eqns ofsort OctetString

     VCDUHEC = AddFront(Octet(0,0,0,0,0,0,0,0),
                       AddFront(Octet(0,0,0,0,0,0,0,0), NullOS)) ;

     NoVCDUHEC = NullOS ;
endtype
```

4.2.2 Data Zones

4.2.2.1 VCDU Insert Zone

The VCDU Insert Zone is represented by the OctetString ADT (5.7).

4.2.2.2 VCDU Data Unit Zone

The VCDU Data Unit Zone is represented by the OctetString ADT.

4.2.3 VCDU Trailer

The VCDU Trailer is an optional component of the VCDU. Its presence or absence and internal configuration are prespecified for a particular Virtual Channel by management. If present, it provides a mechanism for inserting an ‘Operational Control’ field, and/or a ‘VCDU Error Control’ field into the trailing octets of a particular VCDU.

```

type      VCDUTrailer is OctetString
sorts     VCDUTrailer
opns      MakeVCDUTrailer      : OctetString, OctetString -> VCDUTrailer
          NullTrailer         : -> VCDUTrailer
          GetOCF               : VCDUTrailer -> OctetString
          GetECF               : VCDUTrailer -> OctetString
          ConvertVCDUTrailerToOS : VCDUTrailer -> OctetString
eqns      forall OCF1 : OctetString,
          ECF1 : OctetString, Trailer : VCDUTrailer

          ofsort OctetString

          GetOCF(MakeVCDUTrailer(OCF1,ECF1))          = OCF1 ;
          GetECF(MakeVCDUTrailer(OCF1,ECF1))          = ECF1 ;
          ConvertVCDUTrailerToOS(Trailer)              = Append(GetECF(Trailer),
                                                                GetOCF(Trailer)) ;

          ofsort VCDUTrailer

          NullTrailer              = MakeVCDUTrailer(NullOS, NullOS) ;

endtype
    
```

4.2.3.1 Operational Control Field

The Operational Control field allows a Project organization to support a hybrid configuration whereby a Conventional CCSDS system may be operated in conjunction with an Advanced Orbiting System. If present, this 32-bit field shall contain a Command Link Control Word (CLCW).

Note – Actual coding is not represented here; zeros are used instead.

```

type VCDUOperationalControlField is OctetString
opns VCDUOCF : -> OctetString
      NoOCF   : -> OctetString
eqns ofsort OctetString

      VCDUOCF = AddFront(Octet(0,0,0,0,0,0,0,0),
                          AddFront(Octet(0,0,0,0,0,0,0,0),
                                    AddFront(Octet(0,0,0,0,0,0,0,0),
                                              AddFront(Octet(0,0,0,0,0,0,0,0),
                                                        NullOS)))) ;

      NoOCF = NullOS ;
endtype
    
```

4.2.3.2 VCDU Error Control Field

The VCDU Error Control field contains a 16-bit cyclic redundancy code which provides a capability for detecting errors that may have been introduced into VCDUs that have been transmitted without the protection of Reed-Solomon outer coding.

Note – Actual coding is not represented here; zeros are used instead.

```
type VCDUErrorControlField is OctetString
opns VCDUECF      : -> OctetString
     NoECF       : -> OctetString
eqns ofsort OctetString

     VCDUECF = AddFront(Octet(0,0,0,0,0,0,0,0),
                       AddFront(Octet(0,0,0,0,0,0,0,0), NulLOS)) ;

     NoECF = NulLOS ;
endtype
```

4.2.4 Reed-Solomon Check Symbols Field

The Reed-Solomon Check Symbols field contains Reed-Solomon check symbols. A VCDU which has this field appended becomes known as a Coded Virtual Channel Data Unit (CVCDU).

Note – Actual coding is not represented here; zeros are used instead.

```

type ReedSolomonCheckSymbols is OctetString, NaturalNumber
opns GenerateRS : Nat -> OctetString
eqns forall Count : Nat

    ofsort OctetString

        Count Eq 0 =>
            GenerateRS(Count) = NullOS ;

        Count Ne 0 =>
            GenerateRS(Count) = AddFront(Octet(0,0,0,0,0,0,0,0),
                                           GenerateRS(Pred(Count))) ;
endtype
    
```


4.3 Channel Access Data Unit

The Channel Access Data Unit (CADU) consists of a VC_PDU (i.e., a VCDU or a CVCDU, possibly exclusively ORed with a bit transition generator) that is prefixed by a Synchronization Marker. The CADU is represented by the BitString ADT (5.6).

4.3.1 Synchronization Marker

The standard CCSDS Attached Synchronization Marker is a fixed 32-bit pattern that may be represented as '1ACFFC1D' in hexadecimal notation.

```
type SynchronizationMarker is OctetString
opns SyncMarker : -> OctetString
eqns ofsort OctetString

    SyncMarker = AddFront(Octet(0,0,0,1,1,0,1,0),
        AddFront(Octet(1,1,0,0,1,1,1,1),
            AddFront(Octet(1,1,1,1,1,1,0,0),
                AddFront(Octet(0,0,0,1,1,1,0,1), NullOS)))) ;

endtype
```

5 LOTOS DATA TYPES

5.1 BOOLEAN DEFINITION

```

type      Boolean is
sorts     Bool
opns      True, False           : -> Bool
          Not                    : Bool -> Bool
          _And_, _Or_, _Xor_, _Implies_,
          _Iff_, _Eq_, _Ne_      : Bool, Bool -> Bool

eqns      forall x, y : Bool

          ofsort Bool

          Not(True)              = False;
          Not(False)             = True;
          x And True              = x;
          x And False            = False;
          x Or True               = True;
          x Or False             = x;
          x Xor y                 = (x And Not(y)) Or
                                   (y And Not(x));
          x Implies y             = y Or Not(x);
          x Iff y                 = (x Implies y) And
                                   (y Implies x);
          x Eq y                  = x Iff y;
          x Ne y                  = x Xor y;

endtype

```

5.2 BASIC NATURAL NUMBER TYPE

The standard library definition of Basic Natural Number, with constants defined for 2 through 8, is as follows:

```

type      BasicNaturalNumber is
sorts    Nat
opns     0,1,2,3,4,5,6,7,8      : -> Nat
         Succ                    : Nat -> Nat
         Pred                     : Nat -> Nat
         _+_ , _-_, _*_ , _**_   : Nat, Nat -> Nat

eqns     forall m, n: Nat

         ofsort Nat

         m + 0                    = m;
         m + Succ(n)              = Succ(m) + n;

         m - 0                    = m;
         m - Succ(n)              = Pred(m) - n;

         m * 0                    = 0;
         m * Succ(n)              = m + (m * n);

         m ** 0                   = Succ(0);
         m ** Succ(n)             = m * (m ** n);

         Pred(0)                  = 0;
         Pred(Succ(m))            = m;

         1                        = succ(0) ;
         2                        = succ(1) ;
         3                        = succ(2) ;
         4                        = succ(3) ;
         5                        = succ(4) ;
         6                        = succ(5) ;
         7                        = succ(6) ;
         8                        = succ(7) ;

endtype

```

5.3 NATURAL NUMBER

```
type NaturalNumber is BasicNaturalNumber, Boolean
opns  _Eq_, _Ne_, _Lt_,
      _Le_, _Ge_, _Gt_      : Nat, Nat -> Bool

eqns  forall m, n: Nat

      ofsort Bool

      0 Eq 0                = True;
      0 Eq Succ(m)         = False;
      Succ(m) Eq 0         = False;
      Succ(m) Eq Succ(n)   = m Eq n;

      m Ne n                = Not(m Eq n);

      0 Lt 0                = False;
      0 Lt Succ(n)         = True;
      Succ(n) Lt 0         = False;
      Succ(m) Lt Succ(n)   = m Lt n;

      m Le n                = (m Lt n) Or (m Eq n);
      m Ge n                = Not(m Lt n);
      m Gt n                = Not(m Le n);

endtype
```

5.4 BIT

```

type      Bit is NaturalNumber, Boolean
sorts     Bit
opns      0, 1, Bit0, Bit1          : -> Bit
          _Eq_, _Ne_                : Bit, Bit -> Bool
          NatNum                     : Bit -> Nat
eqns      forall x, y: Bit

          ofsort Bit

          Bit0                        = 0 ;
          Bit1                        = 1 ;

          ofsort Nat

          NatNum(0)                   = 0 ;
          NatNum(1)                   = Succ(0) ;

          ofsort Bool

          x Eq y                       = NatNum(x) Eq NatNum(y) ;
          x Ne y                       = NatNum(x) Ne NatNum(y) ;

endtype

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

5.5 OCTET

```
type      Octet is Bit, Boolean, NaturalNumber, BitString
sorts    Octet
opns     Octet
          : Bit, Bit, Bit, Bit, Bit,
          Bit, Bit, Bit -> Octet
          Bit1, Bit2, Bit3, Bit4,
          Bit5, Bit6, Bit7, Bit8
          : Octet -> Bit
          _Eq_, _Ne_
          : Octet, Octet -> Bool
          NatNum
          : Octet -> Nat
          ConvertOtoBS
          : Octet -> BitString

eqns     forall b1, b2, b3, b4, b5, b6, b7, b8: Bit,
          O1, O2: Octet

          ofsort Bit

          Bit1(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b1 ;
          Bit2(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b2 ;
          Bit3(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b3 ;
          Bit4(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b4 ;
          Bit5(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b5 ;
          Bit6(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b6 ;
          Bit7(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b7 ;
          Bit8(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) = b8 ;

          ofsort Bool

          O1 Eq O2
          = (((((((Bit1(O1) Eq Bit1(O2)) And
                    (Bit2(O1) Eq Bit2(O2))) And
                    (Bit3(O1) Eq Bit3(O2))) And
                    (Bit4(O1) Eq Bit4(O2))) And
                    (Bit5(O1) Eq Bit5(O2))) And
                    (Bit6(O1) Eq Bit6(O2))) And
                    (Bit7(O1) Eq Bit7(O2))) And
                    (Bit8(O1) Eq Bit8(O2))) ;

          O1 Ne O2
          = not(O1 Eq O2) ;

          ofsort Nat

          NatNum(Octet(b1,b2,b3,b4,b5,b6,b7,b8)) =
            ((((((((((NatNum(b1) * 2) + NatNum(b2)) * 2) +
                      NatNum(b3)) * 2) + NatNum(b4)) * 2) +
                      NatNum(b5)) * 2) + NatNum(b6)) * 2) +
                      NatNum(b7)) * 2) + NatNum(b8) ;

          ofsort BitString

          ConvertOtoBS(Octet(b1,b2,b3,b4,b5,b6,b7,b8))
          = AddMSB(b1, AddMSB(b2,
                               AddMSB(b3, AddMSB(b4,
                                               AddMSB(b5, AddMSB(b6,
                                                               AddMSB(b7, AddMSB(b8, NullBS)))))))) ;

endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
type CounterOctet is Octet
opns Next      : Octet -> Octet
eqns forall b1,b2,b3,b4,b5,b6,b7 : Bit

    ofsort Octet

Next(Octet(b1,b2,b3,b4,b5,b6,b7,0)) = Octet(b1,b2,b3,b4,b5,b6,b7,1) ;
Next(Octet(b1,b2,b3,b4,b5,b6,0,1)) = Octet(b1,b2,b3,b4,b5,b6,1,0) ;
Next(Octet(b1,b2,b3,b4,b5,0,1,1)) = Octet(b1,b2,b3,b4,b5,1,0,0) ;
Next(Octet(b1,b2,b3,b4,0,1,1,1)) = Octet(b1,b2,b3,b4,1,0,0,0) ;
Next(Octet(b1,b2,b3,0,1,1,1,1)) = Octet(b1,b2,b3,1,0,0,0,0) ;
Next(Octet(b1,b2,0,1,1,1,1,1)) = Octet(b1,b2,1,0,0,0,0,0) ;
Next(Octet(b1,0,1,1,1,1,1,1)) = Octet(b1,1,0,0,0,0,0,0) ;
Next(Octet(0,1,1,1,1,1,1,1)) = Octet(1,0,0,0,0,0,0,0) ;
Next(Octet(1,1,1,1,1,1,1,1)) = Octet(0,0,0,0,0,0,0,0) ;
endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

5.6 BITSTRING

```
type BitString is Bit, NaturalNumber, Boolean
sorts BitString
opns
  NullBS          : -> BitString
  AddLSB          : Bit, BitString -> BitString
  AddMSB          : Bit, BitString -> BitString
  GetLSB          : BitString -> Bit
  GetMSB          : BitString -> Bit
  RemoveMSB       : BitString -> BitString
  Append          : BitString, BitString -> BitString
  LengthOf        : BitString -> Nat
  StripBits       : BitString, Nat -> BitString
  RetainBits      : BitString, Nat -> BitString
  _Eq_            : BitString, BitString -> Bool
  _Ne_            : BitString, BitString -> Bool

eqns
  forall B1, B2 : Bit,
         BS1, BS2 : BitString,
         Nat1, BitLen1 : Nat

  ofsort Bit

  GetLSB(NullBS)          = 0 ;
  GetLSB(AddMSB(B1, NullBS)) = B1 ;
  GetLSB(AddMSB(B1, AddMSB(B2, BS1))) = GetLSB(AddMSB(B2, BS1)) ;

  GetMSB(NullBS)          = 0 ;
  GetMSB(AddMSB(B1, NullBS)) = B1 ;
  GetMSB(AddMSB(B1, BS1))   = B1 ;

  ofsort BitString

  RemoveMSB(NullBS)          = NullBS ;
  RemoveMSB(AddMSB(B1, BS1)) = BS1 ;

  Append(NullBS, BS1)          = BS1;
  Append(BS1, NullBS)          = BS1;
  Append(AddMSB(B1, BS1),
        AddMSB(B2, BS2))      = AddMSB(B2, Append(AddMSB(B1, BS1), BS2)) ;
  AddLSB(B1, NullBS)          = AddMSB(B1, NullBS) ;
  AddLSB(B1, BS1)              = Append(AddMSB(B1, NullBS), BS1) ;

  StripBits(BS1, 0)           = BS1 ;
  StripBits(AddMSB(B1, BS1), succ(Nat1)) = StripBits(BS1, Nat1) ;

  RetainBits(BS1, 0)           = BS1 ;
  RetainBits(AddMSB(B1, BS1), Succ(Nat1)) = AddMSB(B1,
        RetainBits(BS1, Nat1)) ;

  ofsort Nat

  LengthOf(NullBS)          = 0;
  LengthOf(AddMSB(B1, BS1)) = Succ(LengthOf(BS1));
```


CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
ofsort Bool

NullBS Eq NullBS                = True ;

AddMSB(B1,NullBS) Eq
NullBS                          = False ;

NullBS Eq
AddMSB(B1,NullBS)              = False ;

AddMSB(B1,BS1) Eq
NullBS                          = False ;

NullBS Eq
AddMSB(B1,BS1)                 = False ;

AddMSB(B1, BS1) Eq
AddMSB(B2, BS2)                = (B1 Eq B2) And
                                (BS1 Eq BS2) ;

BS1 Ne BS2                      = Not(BS1 Eq BS2) ;

endtype
```

5.7 OCTETSTRING

The following defines OctetString as used in this specification. OctetString is an ordered set of data which contains octets. As a result, octets can be considered as ordered elements in OctetString.

```

type OctetString is Octet, NaturalNumber, BitString, Boolean
sorts OctetString
opns  Nullos      : -> OctetString
      AddFront   : Octet, OctetString -> OctetString
      First      : OctetString -> Octet
      Last       : OctetString -> Octet
      Nth        : OctetString, Nat -> Octet
      Append     : OctetString, OctetString -> OctetString
      ConvertOSToBS : OctetString -> BitString
      ConvertBSToOS : BitString -> OctetString
      StripOctets  : OctetString, Nat -> OctetString
      RetainOctets : OctetString, Nat -> OctetString
      LengthOf    : OctetString -> Nat
      ConvertOSToNat : OctetString -> Nat
      _Eq_, _Ne_   : OctetString, OctetString -> Bool

eqns  forall OS1, OS2 : OctetString,
      O1, O2 : Octet,
      N : Nat,
      BS1 : BitString

      ofsort Octet

      Last(AddFront(O1, Nullos))           = O1 ;
      Last(AddFront(O1, AddFront(O2, OS1))) = Last(AddFront(O2, OS1)) ;

      First(AddFront(O1, Nullos))          = O1 ;
      First(AddFront(O1, OS1))            = O1 ;

      Nth(OS1, 0)                          = First(OS1) ;
      Nth(OS1, succ(0))                    = First(OS1) ;
      Nth(AddFront(O1, OS1), Succ(N))      = Nth(OS1, N) ;

      ofsort OctetString

      Append(OS1, Nullos)                   = OS1 ;
      Append(Nullos, OS1)                   = OS1 ;
      Append(AddFront(O1, OS1),
             AddFront(O2, OS2)) = AddFront(O2, Append(AddFront(O1, OS1),
             OS2)) ;

      StripOctets(OS1, 0)                   = OS1 ;
      StripOctets(AddFront(O1, OS1), Succ(0)) = OS1 ;
      StripOctets(AddFront(O1, OS1), Succ(N)) = StripOctets(OS1, N) ;

      RetainOctets(OS1, 0)                  = Nullos ;
      RetainOctets(AddFront(O1, OS1), Succ(0)) = AddFront(O1, Nullos) ;
      RetainOctets(AddFront(O1, OS1), Succ(N)) = AddFront(O1,
             RetainOctets(OS1, N)) ;

```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
ConvertBStoOS(NullBS) = NullOS ;
ConvertBStoOS(BS1) = AddFront(
    Octet(
        GetMSB(BS1),
        GetMSB(RemoveMSB(BS1)),
        GetMSB(RemoveMSB(RemoveMSB(BS1))),
        GetMSB(RemoveMSB(RemoveMSB(
            RemoveMSB(BS1)))),
        GetMSB(RemoveMSB(RemoveMSB(
            RemoveMSB(RemoveMSB(BS1))))),
        GetMSB(RemoveMSB(RemoveMSB(
            RemoveMSB(RemoveMSB(
                RemoveMSB(BS1)))))),
        GetMSB(RemoveMSB(RemoveMSB(
            RemoveMSB(RemoveMSB(
                RemoveMSB(RemoveMSB(
                    RemoveMSB(BS1)))))))))
    ),
    ConvertBStoOS(StripBits(BS1, 8))
) ;
```

ofsort BitString

```
ConvertOSToBS(NullOS) = NullBS ;
ConvertOSToBS(AddFront(O1, OS1)) = Append(ConvertOSToBS(OS1),
    ConvertOSToBS(O1)) ;
```

ofsort Nat

```
LengthOf(NullOS) = 0 ;
LengthOf(AddFront(O1, OS1)) = Succ(LengthOf(OS1)) ;

ConvertOSToNat(NullOS) = 0 ;
ConvertOSToNat(
    AddFront(O1, NullOS)) = NatNum(O1) ;
ConvertOSToNat(
    AddFront(O1, OS1)) = (NatNum(O1) *
    ((2 ** 8) **
    LengthOf(OS1))) +
    ConvertOSToNat(OS1) ;
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
ofsort Bool

NullOS Eq NullOS                = True ;

AddFront(O1,NullOS) Eq
NullOS                          = False ;

NullOS Eq
AddFront(O1,NullOS)            = False ;

AddFront(O1,OS1) Eq
NullOS                          = False ;

NullOS Eq
AddFront(O1,OS1)              = False ;

AddFront(O1, OS1) Eq
AddFront(O2, OS2)              = (O1 Eq O2) and
                                (OS1 Eq OS2) ;

OS1 Ne OS2                      = not(OS1 Eq OS2) ;

endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
type FillData is OctetString, NaturalNumber
opns MakeFillData : OctetString, Nat -> OctetString
eqns forall OS    : OctetString, N : Nat

  ofsort OctetString

    LengthOf(OS) Ge N =>
  MakeFillData(OS, N)      = RetainOctets(OS, N) ;

    LengthOf(OS) Lt N =>
  MakeFillData(OS, N)      = MakeFillData(Append(OS, OS), N) ;
endtype
```

CCSDS HISTORICAL DOCUMENT

RECOMMENDATION FOR ADVANCED ORBITING SYSTEMS

```
type BitFillData is BitString, NaturalNumber
opns MakeBitFillData : BitString, Nat -> BitString
eqns forall BS : BitString, N : Nat

  ofsort BitString

    LengthOf(BS) Ge N =>
      MakeBitFillData(BS, N)          = RetainBits(BS, N) ;

    LengthOf(BS) Lt N =>
      MakeBitFillData(BS, N)          = MakeBitFillData(Append(BS, BS), N) ;
endtype
```