The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

HIGH PERFORMANCE RELIABILITY PROTOCOL (HPRP)

DRAFT EXPERIMENTAL SPECIFICATION

CCSDS 000.0-W-0

WHITE BOOK

February 2024

AUTHORITY

Issue:	White Book, Issue 0
Date:	January 2024
Location:	Not Applicable

(WHEN THIS EXPERIMENTAL SPECIFICATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

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 documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

CCSDS Secretariat
Space Communications and Navigation Office, 7L70
Space Operations Mission Directorate
NASA Headquarters
Washington, DC 20546-0001, USA

FOREWORD

[Foreword text specific to this document goes here. The text below is boilerplate.]

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CCSDS shall not be held responsible for identifying any or all such patent rights.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Experimental Specification is therefore subject to CCSDS document management and change control procedures, which are defined in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

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

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Experimental Specification. Its ‘White Book’ status indicates that its contents are not stable, and several iterations resulting in substantial technical changes are likely to occur before it is considered to be sufficiently mature to be released for review by the CCSDS Agencies.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document’s technical content.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

DOCUMENT CONTROL

Document	Title and Issue	Date	Status
CCSDS 000.0-W-0	High Performance Reliability Protocol (HPRP), Proposed Draft Experimental Specification, Issue 0	February 2024	Current draft

CONTENTS

SectionPage

Figure

No table of contents entries found.

Table

No table of contents entries found.

1 INTRODUCTION

1.1 PURPOSE

The purpose of this Experimental Specification is to specify the High Performance Reliability Protocol (HPRP), a link layer protocol for transmission of blocks of data designed as an alternative to CCSDS LTP. This standard is to be used over space-to-ground, ground-to-space or space-to-space communications links by space missions.

1.2 SCOPE

This Experimental Specification defines HPRP in terms of:

a) the services provided to the users of this specification;

b) PDU formats for the HPRP protocol; and

c) the procedures required to implement the HPRP Standard

It does not specify:

a) individual implementations or products;

b) the methods or technologies required to perform the procedures; or

c) the management activities required to configure and control the system.

1.3 APPLICABILITY

This Experimental Specification applies to the creation of Agency standards and to the future data communications over space links between CCSDS Agencies in cross-support situations. This Experimental Specification includes comprehensive specification of the data formats and procedures for inter-Agency cross support. It is neither a specification of, nor a design for, real systems that may be implemented for existing or future missions.

The Experimental Specification specified in this document is to be invoked through the normal standards programs of each CCSDS Agency, and is applicable to those missions for which cross support based on capabilities described in this Experimental Specification is anticipated. Where mandatory capabilities are clearly indicated in sections of this Experimental Specification, they must be implemented when this document is used as a basis for cross support. Where options are allowed or implied, implementation of these options is subject to specific bilateral cross support agreements between the Agencies involved.

1.4 RATIONALE

The CCSDS believes it is important to document the rationale underlying the recommendations chosen, so that future evaluations of proposed changes or improvements will not lose sight of previous decisions.

1.5 DOCUMENT STRUCTURE

This document is divided into 6 numbered sections and 1 annex:

1.6 CONVENTIONS AND DEFINITIONS

1.6.1 DEFINITIONS

1.6.1.1 Definitions from the Open System Interconnection (OSI) Basic Reference Model

This Experimental Specification makes use of a number of terms defined in reference [3]. The use of those terms in this Experimental Specification shall be understood in a generic sense; i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

a) Data Link Layer;

b) Physical Layer;

c) service;

d) service data unit.

1.6.1.2 Definitions from OSI Service Definition Conventions

This Experimental Specification makes use of a number of terms defined in reference [4]. The use of those terms in this Experimental Specification shall be understood in a generic sense; i.e., in the sense that those terms are generally applicable to any of a variety of technologies that provide for the exchange of information between real systems. Those terms are:

a) indication;

b) primitive;

c) request;

d) service provider;

e) service user.

1.6.1.3 Definition of Terms Used in This Experimental Specification

For the purposes of this Experimental Specification, the following definitions apply. Many other terms that pertain to specific items are defined in the appropriate sections.

Mission Phase: a period of a mission during which specified communications characteristics are fixed. The transition between two consecutive mission phases may cause an interruption of the communications services.

Physical Channel: a stream of bits transferred over a space link in a single direction.

space link: a communications link between a spacecraft and its associated ground system or between two spacecraft. A space link consists of one or more Physical Channels in one or both directions.

1.6.2 NOMENCLATURE

The following conventions apply for the normative specifications in this Experimental Specification:

a) the words ‘shall’ and ‘must’ imply a binding and verifiable specification;

b) the word ‘should’ implies an optional, but desirable, specification;

c) the word ‘may’ implies an optional specification;

d) the words ‘is’, ‘are’, and ‘will’ imply statements of fact.

NOTE – These conventions do not imply constraints on diction in text that is clearly informative in nature.

1.6.3 INFORMATIVE TEXT

In the normative sections of this document, informative text is set off from the normative specifications either in notes or under one of the following subsection headings:

– Overview;

– Background;

– Rationale;

– Discussion.

1.6.4 CONVENTIONS

In this document, the following convention is used to identify each bit in an *N*-bit field. The first bit in the field to be transmitted (i.e., the most left justified when drawing a figure) is defined to be ‘Bit 0’, the following bit is defined to be ‘Bit 1’, and so on up to ‘Bit *N*-1’. When the field is used to express a binary value (such as a counter), the Most Significant Bit (MSB) shall be the first transmitted bit of the field, i.e., ‘Bit 0’ (see figure 11).

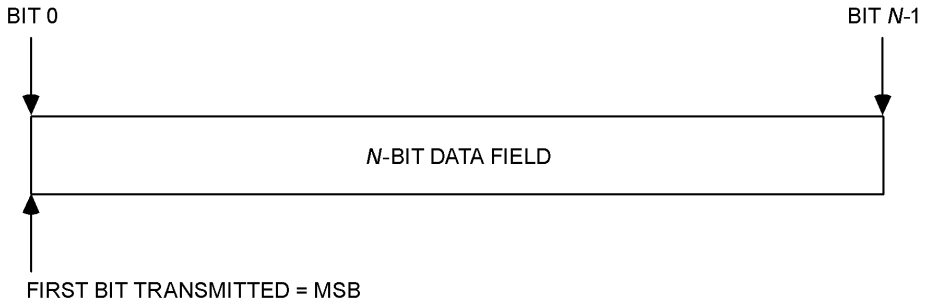

Figure 111-1 Bit Numbering Convention": Bit Numbering Convention

In accordance with standard data-communications practice, data fields are often grouped into 8-bit 'words' which conform to the above convention. Throughout this Experimental Specification, such an 8-bit word is called an 'octet'.

The numbering for octets within a data structure starts with '0'.

1.7 REFERENCES

The following publications contain provisions which, through reference in this text, constitute provisions of this document. At the time of publication, the editions indicated were valid. All publications are subject to revision, and users of this Experimental Specification are encouraged to investigate the possibility of applying the most recent editions of the publications indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.

[Only references required for the implementation of the specification are listed in the References subsection. See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014) for additional information on this subsection.]

2 OVERVIEW

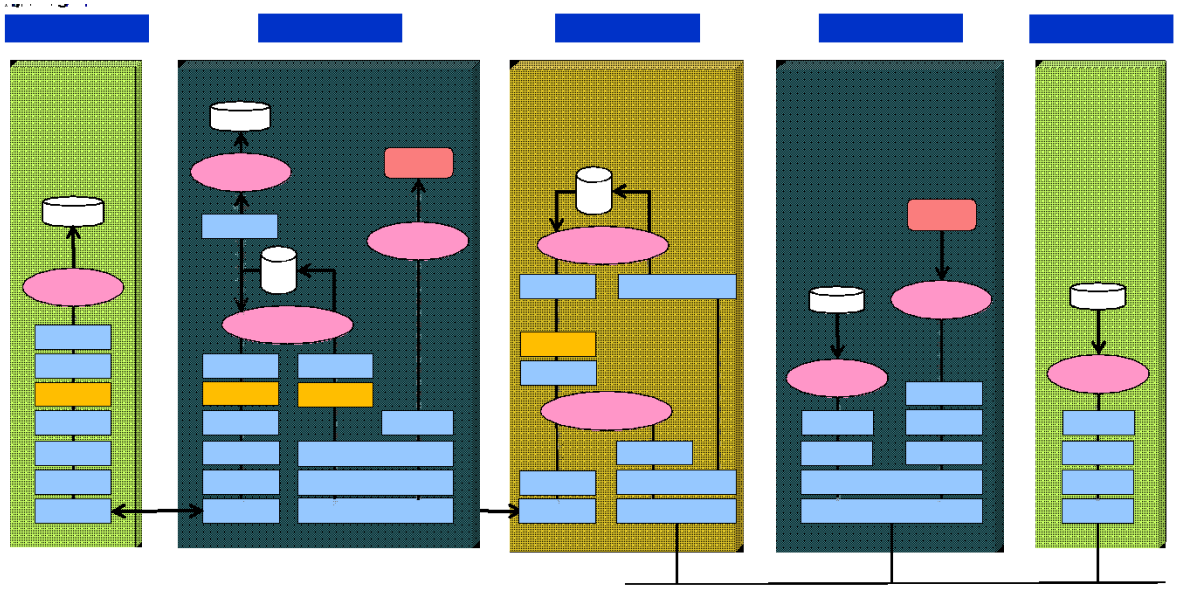

Figure 21 Position of HPRP within SSI ABCBA (HPRP shown in orange)

CCSDS has identified requirements for a protocol to sit between an internetworking protocol such as the Bundle Protocol (reference [E1]) and the various CCSDS data links (see reference [E3]). The High Performance Reliability Protocol (HPRP) sits between the Data Link and the Network Layers of the ISO stack and provides optionally reliable communications over a single data link hop, as shown in Figure 21. HPRP is designed for point-to-point space-to-ground, ground-to-space and space-to-space links; as such, the protocol does not attempt to provide routing/forwarding, instead relying on higher-level protocols to facilitate such functionality. HPRP provides the user with two communication pathways: reliable and unreliable. The unreliable link provides users with a "best-effort" transmission channel, which does not assume the presence of a bi-directional link between communicating entities.

When compared to other ARQ solutions, HPRP provides significant flexibility in communication topologies; first, the protocol may be conveyed over CCSDS data link frames or as packets, as would be provided by the CCSDS Space Packet or Encapsulation Packet Protocols. Additionally, the protocol has been designed to allow different up and downlink channels to be used, providing support for unidirectional RF channels (e.g. Ka-band) or optical links. As the protocol is, by design, tolerant to long delays and significant loss, proper protocol tuning may be used to compensate for mission-specific loss & delay profiles.

These protocol features are supported with self-describing protocol headers, which provides additional flexibility. In a resource-limited implementation of HPRP, certain header lengths may be fixed during the design of the avionics. As the protocol header contains all relevant length fields, a non-resource-constrained implementation can use these length fields to parse data from multiple missions without relying on out-of-band managed parameters. This allows for a subtle performance increase in well-designed implementations; packet lengths can be configured via

out-of-band or “data priming” techniques; incoming packets which match the pre-configured packet lengths would be processed along a fast path, while those which don’t match would lead to a reconfiguration of the header decoding logic (along a slow path), with subsequent packets operating on the newly reconfigured fast path.

The split of protocol data into these multiple pathways also allows for increased throughput in FPGA/ASIC implementations: data packets may be processed through the fast path, while signalling data can be passed to a processor along a slower path. As the protocol signalling logic involves a significant number of branching operations, this allows optimal utilization of both the high-performance yet inflexible FPGA fabric and the slower yet more flexible processor.

2.1 DIFFERENCES TO PREVIOUS LTP STANDARD

While the previous section outlined the fundamental concepts of HPRP, its heritage must also be described. HPRP is based upon concepts of the CCSDS Licklider Transmission Protocol (LTP), albeit with significant modification. To summarize, these tenets are:

- The removal of Self-Delimiting Numeric Values (SDNV)
- The simplification of internal state machines
- The removal of complex and/or largely unused features (such as mixed-color sessions)
- Ease of implementation on non-CPU (ASIC/FPGA) targets
- Extended, harmonized service interface for reliable and unreliable service

The removal of SDNVs has been proposed as a means to ensure determinism within header lengths across a given time/transmission window, which eases implementation within FPGAs and ASICs. This allows all message lengths can be defined *a priori*, though the length of each component may either be signalled via a length parameter within the HPRP header, a numeric profile ID (in which case, lengths will be described in a look-up table), or defined via out-of-band management.

HPRP has removed the notion of “partially-red” blocks, where some data within a single block is sent reliably with eventual re-transmission, while the rest is sent only once. The removal of this feature simplifies the state machine for data reception, which in turn enables an increase in performance. To the best of the authors’ knowledge, although supported within the CCSDS LTP profile, this feature has never been used.

Finally, the segment structure has been simplified to ensure high parsing performance on FPGA’s and ASICs. Instead of using multiple segment types, HPRP header extensions have been utilized to convey protocol signalling and state tracking information.

3 SERVICE DEFINITIONS

3.1 OVERVIEW

This section provides service definition in the form of primitives, which present an abstract model of the logical exchange of data and control information between the protocol entity and the service user. The definitions of primitives are independent of specific implementation approaches.

The parameters of the primitives are specified in an abstract sense and specify the information to be made available to the user of the primitives. The way in which a specific implementation makes this information available is not constrained by this specification. In addition to the parameters specified in this section, an implementation can provide other parameters to the service user (e.g., parameters for controlling the service, monitoring performance, and facilitating diagnosis).

3.2 SERVICES AT THE USER INTERFACE

3.2.1 The following services provided by the protocol shall be made available to the user:

- Initiate a HPRP data block transmission;
- Cancel an ongoing data transfer;
- Receive transfer of data from a remote entity.

3.2.2 Implementations may provide additional services beyond those described in section 4 of this document.

Note: Whether and how such services are invoked is an implementation matter.

3.3 SUMMARY OF PRIMITIVES

3.3.1 The HPRP service shall consume all the following request primitives:

- `Transmission.request`;
- `CancelTransmission.request`;
- `CancelReception.request`.

3.3.2 The HPRP service must deliver the following indication primitives:

- `TransmissionSessionStart.indication`;
- `TransmissionSessionCompletion.indication`;
- `ReceptionSessionCompleted.indication;`
- `SessionCancellation.indication`;

- `SessionTermination.indication`;

3.3.3 The HPRP service may optionally deliver the following indication primitives:

- `RangeReceived.indication;`

3.4 SUMMARY OF PARAMETERS

3.4.1 DESTINATION CLIENT SERVICE ID NUMBER

The client service ID number identifies the layer-(N+1) service to which the block is to be delivered by the receiving HPRP engine that is providing the N-layer service.

3.4.2 SOURCE HPRP ENGINE ID

The Source HPRP engine ID of the HPRP engine that is the transmitter of a data block..

3.4.3 DESTINATION HPRP ENGINE ID

The Destination HPRP engine ID of the HPRP engine that is to be the receiver of the data block.

3.4.4 DATA TO TRANSMIT

Data to transmit represents a length-bounded set of binary data.

3.4.5 RELIABLE TRANSMISSION

Reliable transmission is a Boolean flag which specifies that a session utilizes the reliable transmission/reception mechanisms of HPRP.

3.4.6 SESSION NUMBER

The session number is a unique identifier of a single session.

3.4.7 REASON CODE

Reason Code is an integer that identifies to a remote HPRP engine or user the reason behind a particular action (typically the cancellation of a transmission).

3.4.8 OFFSET

The offset of a byte within a block is the number of bytes that precede it in the block.

3.4.9 LENGTH

Length is the number of octets in a logical group of octets.

3.4.10 RECEIVED DATA

The data delivered by HPRP to a destination HPRP engine.

NOTE – It is assumed that this includes the data itself as well as length information.

3.4.11 RECEPTION MAP

The reception map is an array of (offset, length) fields, representing data which has been successfully received.

3.5 HPRP SERVICE PRIMITIVES

3.5.1 TRANSMISSION.REQUEST

3.5.1.1 Function

The `Transmission.request` primitive shall be used by the user to request delivery of a sequence of bytes to the destination client service via an HPRP engine.

3.5.1.2 Semantics

`Transmission.request` shall provide the following parameters:

```
Transmission.request (destination client service ID,
destination HPRP engine ID,
reliable transmission,
client service data to send)
```

3.5.1.3 When Generated

`Transmission.request` may be generated by the HPRP user at any time.

3.5.1.4 Effect on Receipt

Receipt of a `Transmission.request` shall cause the HPRP engine to attempt the transmission of the data, subject to resource/outbound session limits.

If transmission can be started, receipt of `Transmission.request` results in the delivery of a `TransmissionSessionStart.indication` to the user so that the transmission may be subsequently uniquely identified.

If transmission cannot be started and session data cannot be enqueued, receipt of a `Transmission.request` shall result in the delivery of a `TransmissionSessionTermination.indication`.

3.5.2 CANCELTRANSMISSION.REQUEST

3.5.2.1 Function

The `CancelTransmission.request` primitive shall be used by the user to terminate transmission of a enqueued or in-progress session.

3.5.2.2 Semantics

`CancelTransmission.request` shall provide the following parameters:

```
CancelTransmission.request (destination engine ID,
Session number)
```

3.5.2.3 When Generated

`CancelTransmission.request` may be generated by the HPRP user at any time.

3.5.2.4 Effect on Receipt

Receipt of a `CancelTransmission.request` shall cause the HPRP engine to immediately cancel the session identified by (destination engine ID, Session Number). Once the session is cancelled and resources are released, the HPRP engine shall generate a `sessionCancellation.indication` message.

3.5.3 CANCELRECEPTION.REQUEST

3.5.3.1 Function

The `CancelReception.request` primitive shall be used by a receiving user to terminate reception of an in-progress session from an HPRP engine.

3.5.3.2 Semantics

`CancelReception.request` shall provide the following parameters:

```
CancelReception.request (destination engine ID,
Session number)
```

3.5.3.3 When Generated

`CancelReception.request` may be generated by the HPRP user at any time.

3.5.3.4 Effect on Receipt

Receipt of a `CancelReception.request` shall cause the HPRP engine to immediately cancel the session identified by (destination engine ID, Session Number). Once the session is cancelled and resources are released, the HPRP engine shall generate a `sessionCancellation.indication` message.

3.5.4 TRANSMISSIONSESSIONSTART.INDICATION

3.5.4.1 Function

The `TransmissionSessionStart.indication` primitive shall be generated by a transmitting HPRP engine upon receipt and validation of a `transmission.request`.

The

3.5.4.2 Semantics

`TransmissionSessionStart.indication` shall provide parameters as follows:

```
TransmissionSessionStart.indication (destination HPRP
engine ID,
Session number)
```

3.5.4.3 When Generated

At the sender, a `TransmissionSessionStart.indication` shall be generated by an HPRP engine once the engine has accepted a transmission request from the user.

3.5.4.4 Effect on Receipt

The effect of receipt of a `TransmissionSessionStart.indication` is application-dependent.

3.5.5 TRANSMISSIONSESSIONCOMPLETION.INDICATION

3.5.5.1 Function

The `TransmissionSessionCompletion.indication` primitive shall be generated by a transmitting HPRP engine in response to a successfully-completed HPRP session

3.5.5.2 Semantics

`TransmissionSessionCompletion.indication` shall provide parameters as follows:

```
TransmissionSessionCompletion.indication (destination
HPRP engine ID,
Session number)
```

3.5.5.3 When Generated

At the sender, a `TransmissionSessionCompletion.indication` shall be generated by an HPRP engine once the engine has completed a transmission from the user. For reliable sessions, all re-transmission must be completed prior to generation of this indication, and session closure must have completed. For unreliable sessions, this indication is transmitted by the sending engine as soon as the relevant session closure data has been transmitted.

3.5.5.4 Effect on Receipt

The effect of receipt of a `TransmissionSessionCompletion.indication` is application-dependent.

3.5.6 RECEPTIONSESSIONCOMPLETED.INDICATION

3.5.6.1 Function

The `ReceptionSessionCompletion.indication` primitive shall be generated by a receiving HPRP engine in response to a completed HPRP session.

3.5.6.2 Semantics

`ReceptionSessionCompletion.indication` shall provide parameters as follows:

```
ReceptionSessionCompletion.indication (Source HPRP engine
ID,
Destination HPRP Service ID,
Reliable transmission,
Session number,
(optional) Reception map,
(optional) Received data)
```

3.5.6.3 When Generated

A `ReceptionSessionCompletion.indication` shall be generated by an HPRP engine once the engine has completed reception of a session from a remote engine.

For a given session, users shall not expect to receive any further notifications after reception of a *ReceptionSessionCompletion.indication* primitive.

3.5.6.3.1 Optional Parameters

3.5.6.3.1.1 If the *rangeReceived.indication* primitive is used, the *reception map* and *received data* fields may be omitted.

3.5.6.4 Effect on Receipt

The effect of receipt of a `ReceptionSessionCompletion.indication` is application-dependent.

3.5.7 SESSIONTERMINATION.INDICATION

3.5.7.1 Function

The `sessionTermination.indication` primitive shall be generated by a HPRP engine in response to a cancelled or failed HPRP session

3.5.7.2 Semantics

`sessionTermination.indication` shall provide parameters as follows:

```
sessionCancellation.indication (Source HPRP engine ID,
Destination HPRP engine ID,
Session Number,
Reason Code)
```

3.5.7.3 When Generated

An `sessionTermination.indication` shall be generated by an HPRP engine in the event of a session cancellation, either requested by the user or due to error.

3.5.7.4 Effect on Receipt

The effect of receipt of a `sessionTermination.indication` is application-dependent.

3.5.8 OPTIONAL: RANGERECEIVED.INDICATION

3.5.8.1 Function

The optional `rangeReceived.indication` primitive shall be generated by a receiving HPRP engine in response to the reception of data within a HPRP session. Unlike the *receptionSessionCompleted.indication,* this indication may represent the ad-hoc reception of incomplete data.

3.5.8.2 Semantics

`rangeReceived.indication` shall provide parameters as follows:

```
rangeReceived.indication (Source HPRP engine ID,
Destination HPRP Service ID,
Session number,
Offset,
Length,
received data)
```

3.5.8.3 When Generated

An `rangeReceived.indication` shall be generated by an HPRP engine once the engine has received new data from a remote engine. Optionally, a timeout or a minimum size may be used to perform concatenation prior to the transmission of the indication.

3.5.8.4 Effect on Receipt

The effect of receipt of a `rangeReceived.indication` primitive is application-dependent.

Note: data received by this indication may be out-of-order or include repeats.

3.6 SERVICES REQUIRED FROM THE UNDERLAYING COMMUNICATION LAYER

The service primitives and parameters required to access the services of the underlying communication layer are:

- `TRANSFER.request(segment to transmit)`
- `TRANSFER.indication(received segment)`

Note: the transfer services should perform all concatenation and de-creation of the service. The HPRP protocol does not add any per-segment metadata to describe the size of a given segment, relying on this service. The transfer service should also provide error detection; in the event of a transfer service error, the PDU shall be dropped.

HPRP includes no security mechanisms of any kind but is of course subject to the same confidentiality, data integrity, authentication, and availability concerns as any other protocol. Given the absence of security mechanisms built into HPRP itself, the underlying complete communication stack must provide whichever services are required in order to ensure the secure operation of the protocol as deployed. This might imply application of security at the link layer, the network layer or application layer. The details of these services are necessarily opaque to HPRP and are beyond the scope of this Recommendation.

4 DATA FORMATS

4.1 HPRP SEGMENT

Bit	0 1	2 3	4 5	6 7
0	Version Number	*Extension Control Flags*	Segment Type ID	Unused
		Sys. extensions / User extensions		
8	Session Originator Length		Session Number Length	
16	Session Originator			
VAR	Session Number			
VAR	Header Extensions Length (optional)			
VAR	Header Extensions (Optional, variable)			

4.1.1 VERSION NUMBER

4.1.1.1 Bits 0-1 of the HPRP segment header shall contain the version number field.

4.1.1.2 The version number field must be set to ‘01’ binary.

4.1.2 EXTENSION CONTROL FLAGS

4.1.2.1 Bits 2-3 of the HPRP segment header shall contain the “extension control flags”

4.1.2.2 If the *extensions control flag* bits are set to ‘00’ binary, then no extension headers are present and the *header extensions length* field must not be present.

4.1.2.3 Bit 2 of the HPRP segment header shall contain the “system extensions present” flag.

4.1.2.4 This single-bit flag shall be used to control the processing of HPRP signaling data, as it indicates the presence of HPRP signaling data.

4.1.2.5 Bit 3 of the HPRP segment header shall contain the “user extensions present” flag.

NOTE – This flag indicates the presence of user-specific extension data, not relevant to HPRP protocol processing.

4.1.3 SEGMENT TYPE ID

4.1.3.1 Bits 4-5 of the HPRP Segment Header shall represent the *segment type ID* field.

NOTE – The segment type ID field indicates the type of segment which follows this header, and is used for protocol processing.

4.1.3.2 The segment type ID field must be set to one of the following values:

a. 0x00 (00b) - Reliable Data

b. 0x01 (01b) - Unreliable Data

c. 0x02 (10b) - Extension Container

Note: all segment types may contain extensions.

4.1.4 UNUSED VALUES

4.1.4.1 Bits 6-7 of the HPRP segment header must be set to ‘00’ binary

4.1.5 SESSION ORIGINATOR LENGTH

4.1.5.1 Bits 8-11 of the HPRP segment header shall contain the *session originator length* field.

4.1.5.2 The *session originator length* field must contain the length of the *session originator* value, in octets.

4.1.5.3 The *session originator length* must be set to a value between 1-8 (inclusive).

4.1.6 SESSION NUMBER LENGTH

4.1.6.1 Bits 12-15 of the HPRP segment header shall contain the *session number length* field.

4.1.6.2 The *session number length* field must contain the length of the *session number* field, in octets.

4.1.6.3 The *session number length* must be set to a value between 1-8 (inclusive)

4.1.7 SESSION ORIGINATOR

4.1.7.1 The field starting at bit 16 must represent the session originator. The session originator is defined as the HPRP engine which began the current session.

4.1.7.2 The length of this field (in octets) must be represented by the value of the *session originator length* field.

4.1.7.3 The session originator must be a uniquely identifiable integer.

NOTE - The scope of identifiability is TBD; it may be chosen on a per-agency level or following a SANA registry.

4.1.8 SESSION NUMBER

4.1.8.1 The field immediately following the session originator must represent the session number of this segment, representing the session that this segment belongs to.

4.1.8.2 The length of this field (in octets) must be represented by the value of the *session number length* field.

4.1.8.3 The value *session number length* field must be incremented by one for each session created by this entity.

NOTE – The initial value for the session number may be seeded with the value 0 or a randomly-generated number. It is understood that a sequentially increasing session number is vulnerable to replay attacks, etc; this is not a significant issue in the use-cases for which HPRP is designed for.

4.1.8.4 If a segment is sent in response to a received segment the *session number* must be equal to the received session number.

4.2 HEADER EXTENSIONS

4.2.1 If the *system extensions present* bit is set, then the HPRP entity must process the header extensions.

4.2.2 If the *user extensions present* bit is set, then some header extension sections are not relevant to the decoding logic and may be ignored or passed to the user.

NOTE – This behavior is similar to the Operational Control Field within CCSDS transfer frames.

4.2.3 If both bits are set, then some of the headers are entity-relevant and must be processed by the HPRP entity, and must be inserted into the header extensions area first.

4.2.4 If the *extension control flags* value is set to ‘00’ binary, than no extension data is present and no extension processing shall occur.

4.2.5 HEADER EXTENSIONS LENGTH

4.2.5.1 If either bit of the *extension control flags* is set, then the 8-bit *header extension length* field must follow the session number field, without gap.

4.2.5.2 The *header extensions length* field must be set to the total size, in octets, of the *header extensions* area.

4.2.6 EXTENSION AREA

4.2.6.1 The *header extension* area must immediately follow the *header extension length* field.

4.2.6.2 The length of the *header extension* area (in octets) must equal the length of the *header extension length* field.

4.2.7 EXTENSION HEADER

4.2.7.1 Header extensions must be prefixed with an extension header, shown below:

Bit	0	1	2	3	4	5	6	7
0	Extension Identifier				Extension Serial Number Length			
8	Extension Length							
16	Extension Serial Number							
VAR	Extension Data (optional)							

4.2.7.2 Extension Identifier

4.2.7.2.1 Bit 0-3 of the extension header must represent the *extension identifier*.

4.2.7.2.2 The *extension identifier* must be set to one of the following values:

Extension ID	System Extension	Description	Must be acknowledged
0x00	X	Data Acknowledgement Request	Y
0x01	X	Data Acknowledgement	Y
0x02	X	Session Management	N
0x03	X	Metadata Acknowledgement	Y
0x04-0x0A		Reserved extensions	TBD
0x0B-0x0E		*User-defined Extensions*	
0x0F	X	Header Padding Extension (optional)	

4.2.7.2.3 Each extension identifier must be unique within a given PDU; the protocol does not support multiple extensions with the same ID.

4.2.7.3 Serial Number Length

4.2.7.3.1 Bits 4-7 of the extension header must represent the *serial number length* field.

4.2.7.3.2 The *serial number length* field must contain the length of the *serial number* field, in octets.

4.2.7.3.3 The *serial number length* must be set to a value between 1-8 (inclusive)

4.2.7.4 Extension Length

4.2.7.4.1 Bits 8-15 of the extension header must represent the *extension length* field

4.2.7.4.2 The *extension length* field must be set to the total size, in octets, of the *extension data* area.

4.2.7.4.3 The *extension length* must be set to a value between 1-255 (inclusive).

NOTE – This value is effectively bounded by the maximum length of the extension header area; therefore, the extension length can never be longer than the *header extensions length* minus the length of a single extension header.

4.2.7.5 Extension Serial Number

4.2.7.5.1 The field starting at bit 16 must represent the extension serial number.

4.2.7.5.2 The length of this field (in octets) must be represented by the value of the *serial number length* field.

4.2.7.5.3 The extension serial number must be unique across a given session and extension identifier.

4.2.7.6 Extension Data

4.2.7.6.1 The extension data field must immediately follow the *extension serial number* field, without gaps.

4.2.7.6.2 The length of this field (in octets) must be represented by the value of the *extension length* field.

NOTE – This limits the *extension data* field to a maximum of 255 bytes.

4.2.7.6.3 The *extension data* field must contain extension specific data.

4.2.8 EXTENSION TYPES

4.2.8.1 Data Acknowledgement Request Extension

Bit	0	1	2	3	4	5	6	7
0	Requested report type							

4.2.8.1.1 Header Values

4.2.8.1.1.1 The *extension identifier* field of the extension header must be set to 0x00.

4.2.8.1.1.2 The *session number* field of the HPRP segment header must be set to the session number of the referenced session.

4.2.8.1.2 Acknowledgement

4.2.8.1.2.1 Data Acknowledgement request extensions must be acknowledged by the receiving engine via a metadata acknowledgment message.

4.2.8.1.3 Requested report type

4.2.8.1.3.1 Bit 0-7 of the extension header must represent the *requested report type* field.

4.2.8.1.3.2 The *requested report type field* must be set to 0x00.

Note: other values may be used in the future

4.2.8.2 Data Acknowledgement Extension

Bit	0	1	2	3	4	5	6	7
0	Type							
8	Data Descriptor Length							
16	Reception Claim Count							
VAR	Lower Bound							
Repeating section (n=Reception Claim Count)								
VAR	Claim Offset							
VAR	Claim Length							

4.2.8.2.1 Header Values

4.2.8.2.1.1 The *extension identifier* field of the extension header must be set to 0x01.

4.2.8.2.1.2 The *session number* field of the HPRP segment header must be set to the value of the current session.

4.2.8.2.2 Acknowledgement

4.2.8.2.2.1 Data Acknowledgement extensions must be acknowledged by the receiving engine via a metadata acknowledgment message.

4.2.8.2.3 Type

4.2.8.2.3.1 Bits 0-7 of the data acknowledgment extension header must represent the *type* field.

4.2.8.2.3.2 The *Type* field must be set to one of the following values to describe the source of the acknowledgement

- 0x00 – For Synchronously-transmitted acknowledgements, sent in response to a data acknowledgement request.
- 0x01 – for Asynchronously-transmitted acknowledgements, sent due to implementation-specific heuristics.

4.2.8.2.4 Data Descriptor Length

4.2.8.2.4.1 Bits 8-15 of the header must represent the *Data Descriptor Length* field.

4.2.8.2.4.2 The *data descriptor length* field must contain the length, in octets, of any length or offset field within the data descriptor.

Note – Presently, this value describes the length of the *claim offset*, *claim length*, and *lower bound* fields.

4.2.8.2.4.3 The *data descriptor length* field must be within the range of 1-8, inclusive.

4.2.8.2.5 Reception claim count

4.2.8.2.5.1 Bits 16-23 of the header must represent the *Reception claim count* field.

4.2.8.2.5.2 The *Reception claim count* field must contain the number of *reception claims* within the extension.

4.2.8.2.6 Lower Bound

4.2.8.2.6.1 The field starting at bit 24 of the data segment header must represent the lower bound.

4.2.8.2.6.2 The length, in octets, of the lower bound field must be set to the length provided by the *data descriptor length* field.

4.2.8.2.6.3 The lower bound field represents the highest byte offset (starting from the beginning of the session, byte 0) which has been successfully received and processed within the session.

4.2.8.2.6.4 If the data acknowledgement is the first in a given session, the *lower bound* field must be set to 0.

4.2.8.2.7 Repeating Claim Section

4.2.8.2.7.1 The field immediately following the *lower bound* field is must contain a series of concatenated claims, without gaps.

4.2.8.2.7.2 The repeating claim section must have *repeating claim count* number of entries.

4.2.8.2.7.3 Each claim must be comprised of two values: the *offset* and *length.*

4.2.8.2.7.4 Claim Offset

4.2.8.2.7.4.1 The first field within an individual claim must be the *claim offset* field.

4.2.8.2.7.4.2 The length, in octets, of the *claim offset* field must be provided by the *data descriptor length* field.

4.2.8.2.7.4.3 The offset field must represent the start, in octets, of an un-received claim of data within the given HPRP session.

4.2.8.2.7.5 Claim Length

4.2.8.2.7.5.1 The second field within an individual claim must be the *claim length* field.

4.2.8.2.7.5.2 The length, in octets, of the *claim length* field must be provided by the *data descriptor length* field.

4.2.8.2.7.5.3 The claim length field must represent the length, in octets, of an un-received claim of data within the given HPRP session.

Note: A single unacknowledged span may be calculated as (offset, offset + length)

4.2.8.3 Session Management Extension

Bit	0	1	2	3	4	5	6	7
0	Session Owner?	Reason						

4.2.8.3.1 Header Values

4.2.8.3.1.1 The *extension identifier* field of the extension header must be set to 0x02.

4.2.8.3.1.2 The *session number* field of the HPRP segment header must be set to the value of the current session.

4.2.8.3.2 Acknowledgement

4.2.8.3.2.1 Acknowledgement of session management extensions is optional.

4.2.8.3.3 Session Owner

4.2.8.3.3.1 Bit 0 of the header must represent the *session owner* field.

4.2.8.3.3.2 The *session owner* field must be set to true (1b) if the source HPRP engine is transmitting this extension and false (0b) otherwise.

4.2.8.3.4 Reason

4.2.8.3.4.1 Bits 1-7 of the header must represent the *reason* field.

4.2.8.3.4.2 The *reason* field must be set to one of the values listed in the *Code* column of the table below:

Code	Session Owner	Name	Description
1	0/1	User Cancelled	The user cancelled the session.
2	0/1	System Error	A system-level error occurred and the entity cannot continue processing this session.
3	0	Unreachable	The receiving HPRP entity cannot reach the client service or end-user.
4	1	Retransmission time exceeded	The maximum retransmission time (for metadata or data) was exceeded.
5	1	Retransmission limit exceeded	The maximum number of retransmissions (for metadata or data) were exceeded.
6	0/1	Suspension Requested	The session should be suspended; all timers (on sender and receiver side) should be suspended.
7	1	Session Completed	The last segment of data for this session has been sent alongside this extension.

4.2.8.4 Metadata Acknowledgement Extension

Bit	0	1	2	3	4	5	6	7
0	Metadata acknowledgement count							
8	*Repeating section (n=* Metadata acknowledgement count*)*							
VAR	Extension Identifier ID							
VAR	Extension Serial Number.							

4.2.8.4.1 Header Values

4.2.8.4.1.1 The *extension identifier* field of the extension header must be set to 0x03.

4.2.8.4.1.2 The *session number* field of the HPRP segment header must be set to the value of the current session.

4.2.8.4.2 Metadata Acknowledgement Count

4.2.8.4.2.1 Bits 0-7 of the header must represent the *Metadata Acknowledgment count* field.

4.2.8.4.2.2 The *Metadata Acknowledgment count* field must contain the number of *metadata acknowledgements* within the extension.

4.2.8.4.3 Repeating Section

4.2.8.4.3.1 The field starting at bit 8 must contain a series of concatenated metadata acknowledgments, without gaps.

4.2.8.4.3.2 The repeating section must have *Metadata Acknowledgment count* number of entries.

4.2.8.4.3.3 Each must be comprised of two values: the *extension identifier ID* and *extension serial number.*

4.2.8.4.3.4 Each *extension identifier ID* and *serial number* must be set to the values of a previously-received extension.

4.2.8.4.3.5 Extension Identifier

4.2.8.4.3.5.1 The first field within a single entry must be the *extension identifier*

4.2.8.4.3.5.2 The extension identifier field must be 1 octet long.

4.2.8.4.3.5.3 The extension identifier field must be set to a valid extension type.

Note: Although the extension identifier field within the extension header is 4 bits long, this field is 8 bits. This is an intentional design choice to maintain byte-alignment.

4.2.8.4.3.6 Extension Serial Number

4.2.8.4.3.6.1 The second field within a single entry must be the *extension serial number*

4.2.8.4.3.6.2 The length of this field (in octets) must be represented by the value of the serial number length field within the extension header.

4.2.8.4.3.6.3 The extension serial number field must be set to the value of a previously-received serial number.

4.3 DATA SEGMENTS

Bit	0	1	2	3	4	5	6	7
0	Client Service ID Length				Data Descriptor Length			
8	Client Service ID							

VAR	Offset
VAR	Block Length
VAR	*Data*

4.3.1 CLIENT SERVICE ID LENGTH

4.3.1.1.1 Bits 0-3 of the data segment header must represent the *client service ID length*

4.3.1.1.2 The *client service ID length* field must contain the length of the *client service ID* field, in octets.

4.3.1.1.3 The *client service ID length* must be set to a value between 1-8 (inclusive)

4.3.2 DATA DESCRIPTOR LENGTH

4.3.2.1.1 Bits 4-8 of the data segment header must represent the *data descriptor length*

4.3.2.1.2 The *data descriptor length* field must contain the lengths of the *offset* and *length* fields, in octets.

4.3.2.1.3 The *data descriptor length* must be set to a value between 1-8 (inclusive).

4.3.2.2 **Client Service ID**

4.3.2.2.1 The field starting at bit 9 of the data segment header must represent the client service ID.

4.3.2.2.2 If no client service ID is required, the client service ID must be set to ‘0’.

Note: The field must still be present, even if set to zero. This simplifies parallel implementations (e.g. on FPGA’s and ASIC’s)

4.3.2.2.3 Service ID’s must remain constant across a single session.

4.3.3 OFFSET

4.3.3.1.1 The offset field must immediately follow the client service ID field, without gaps.

4.3.3.1.2 The offset field must represent the “position” of the data within the HPRP block in octets.

4.3.4 BLOCK LENGTH

4.3.4.1.1 The *block length* field must immediately follow the *offset* field, without gaps.

4.3.4.1.2 The *block length* field must represent the total amount of data which was presented at the transmitting engine for transmission within the current session.

4.3.4.1.3 If the total length of data within a session is unknown, the *length* field must be set to ‘0’.

4.3.4.1.4 The *block length* field must remain constant for the duration of a session.

Note: The block length field is provided for two reasons; first, it allows the receiving HPRP engine to allocate a contiguous area of memory for a given session, which increases performance. Second, it facilitates deterministic allocation in spite of out-of-order arrivals. Regardless of which data segment is received first, the correct buffer sizing can be performed.

4.3.5 DATA

4.3.5.1 Data must be provided as a byte array, without padding, etc.

4.3.5.2 The maximum length of the data field shall be defined in the MIB.

Note: The maximum length of the data field may be limited by the underlying transport layer. The MIB needs to be configured accordingly. Different maximum length values could be chosen for different transport layers or different remote engines.

4.4 EXTENSION CONTAINER SEGMENT

4.4.1 Extension data must immediately follow the HPRP header, without padding or additional headers.

4.4.2 The entire capacity of the frame may be occupied with extension data.

Note: the capacity may be limited by the underlying link-layer protocol or MIB.

4.4.3 If more than 255 octets of extension information are to be included, the extension length field must be set to *0*

4.4.4

5 PROTOCOL PROCEDURES

5.1 OVERVIEW

This section describes the internal procedures that are triggered by events, such as user-driven services and/or internal protocol state transitions.

5.2 PROCEDURES AT THE SENDING END

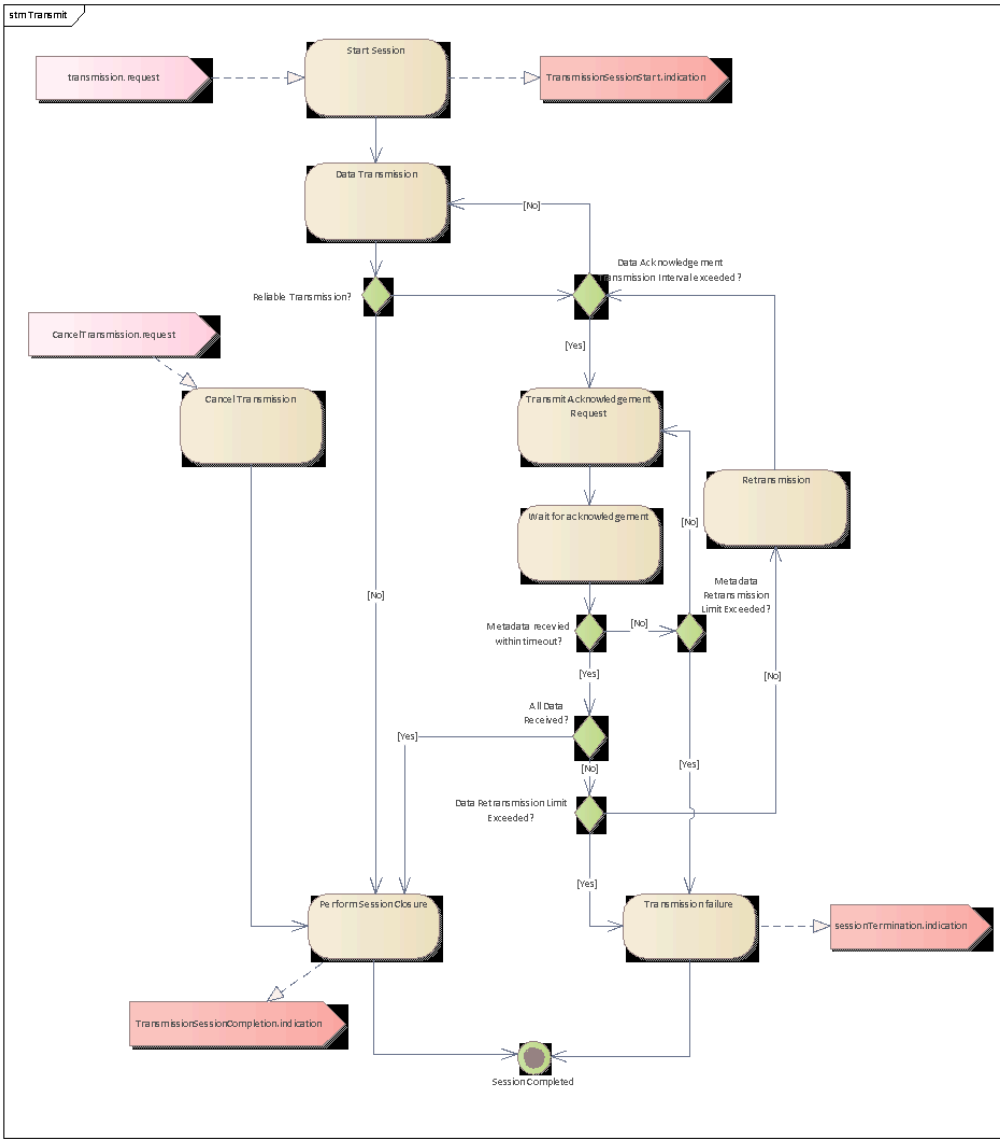

Figure 51 Transmission Procedures

5.2.1 START SESSION

This state performs all transmitter-side housekeeping required to start a new session. Depending on the configuration of the HPRP engine, this may include any relevant pre-buffering of data, as well as the resetting of all retransmission timers & counters.

HPRP does not have a specific “session start” marker; therefore, nothing must be specifically transmitted to indicate the start of a session.

5.2.2 CANCEL TRANSMISSION

If triggered by a *cancelTransmission.indication*, this procedure shall release all resources and transition to the *perform session closure* procedure, transmitting a session management extension with the SESSION_CANCELLED reason code.

5.2.3 DATA TRANSMISSION

This state is responsible for the creation of new data segments (§4.3), and enqueueing them to the (N-1) layer, fragmenting to the maximum data segment size (as required). If the final data segment is less than the data segment size, no padding shall be inserted. Upon the end of reception, this procedure triggers the *Reliable Transmission?* trigger, thus triggering the retransmission loop.

Note: Depending on implementation, this may also maintain responsibility for requests from storage, etc.

5.2.4 TRANSMIT ACKNOWLEDGEMENT REQUEST

This procedure is triggered by the *Data Acknowledgement Transmission Interval exceeded?* checkpoint, which must trigger upon one of the following conditions:

- Whether the current number of bytes transmitted since the last *data acknowledgement extension* exceeds the threshold set by the *Acknowledgement Transmission Interval* managed parameter.
- Whether the time since last data acknowledgement extension transmission exceeds the value provided by the *Acknowledgement Transmission Interval* managed parameter.
- Reception of the external trigger provided by the *Data Transmission* procedure, then and trigger the data acknowledgment process as required.

As outlined in section 5.1.2.1.1, data acknowledgement within HPRP is signalled by the creation of data acknowledgement extensions (§4.2.8.2). This request may be transmitted within a unique segment (e.g. within a EXTENSION_CONTAINER) or as a header extension of a data segment.

5.2.5 WAIT FOR ACKNOWLEDGEMENT

Waits for reception of a data acknowledgement encapsulated within a data acknowledgement extension (4.2.8.2). This procedure shall be guarded by a managed timeout parameter. A single

session may transition to this state multiple times within a single session, due to retransmission(s).

If a data acknowledgment (and associated acknowledgement of the transmitted extension) were received within the timeout window, this procedure may be terminated. Otherwise, the acknowledgment request must be retransmitted, subject to the *metadata retransmission limit*.

If all data has been received, this procedure may be terminated and the session can skip to the *perform session closure* procedure.

If unacknowledged data remains, the sending engine shall test whether the *data retransmission limit* parameter has been exceeded. If so, the session may progress to the *transmission failure* state.

5.2.6 RETRANSMISSION

Retransmits (1:*N*) unreceived spans of data, based upon the claims provided in the previous data acknowledgement.

This procedure must, for each unreceived span, begin transmission from the *offset* and transmit until the *offset* + *length* of data, fragmenting to the maximum data segment size (as required). If the final data segment is less than the data segment size, no padding shall be inserted.

5.2.7 PERFORM SESSION CLOSURE

If triggered by the acknowledgement of data completeness or (for unreliable sessions) the complete transmission of data from the sending engine, transmits a sessionManagement extension with the SESSION_COMPLETED reason code (§4.2.8.3).

For sessions which have been cancelled, , transmits a sessionManagement extension with the SESSION_CANCELLED reason code (§4.2.8.3).

After transmission, all resources must be released and no further segments shall be transmitted or received.

Note: the specific shutdown processes which must be followed are implementation-dependent.

5.2.8 TRANSMISSION FAILURE

Transmits a sessionManagement extension with a failure reason code (per §4.2.8.3) and prepares to release all resources.

Note: the specific shutdown processes which must be followed are implementation-dependent.

5.3 PROCEDURES AT THE RECEIVING END

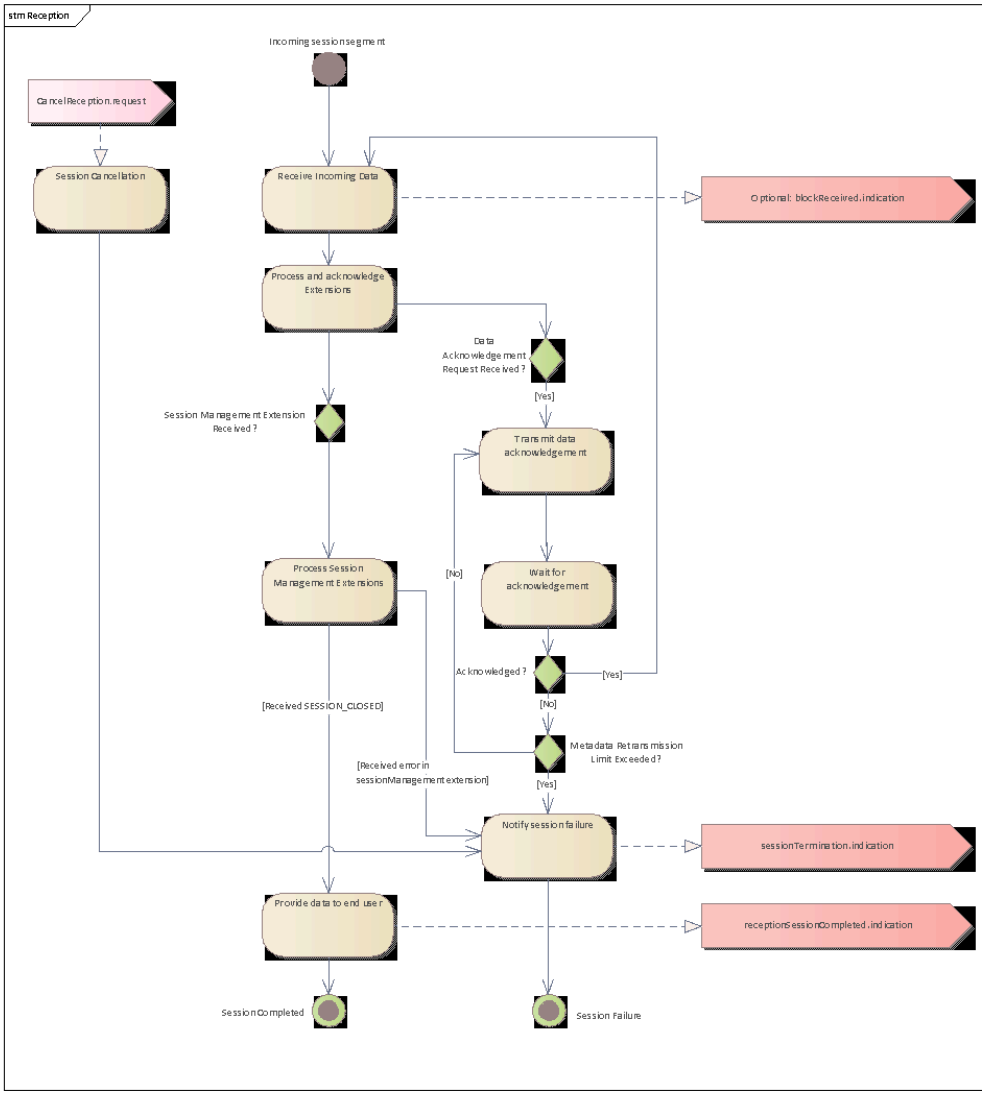

Figure 52 Reception Procedures

5.3.1 RECEIVE INCOMING DATA

This procedure receives incoming data & extension container segments, parsing all system extensions. Buffering behaviour within this procedure is entirely dependent upon whether the user wishes to use the *rangeReceived.indication* primitive. If that primitive is used, client data contained within data segments shall be provided as soon as it arrives, although small buffers may be used to provide contigious ranges of data to the user.

If the user expects the *receptionSessionCompleted.indication* primitive, this procedure must buffer incoming data.

Note: the details of buffering is implementation-specific.

5.3.2 SESSION CANCELLATION

When triggered by a *cancelReception.request* primitive, terminates all timers and transitions to the *Notify Session Failure* procedure, transmitting a sessionManagement extension with a SESSION_CANCELLED reason code.

5.3.3 PROCESS AND ACKNOWLEDGE EXTENSIONS

Incoming extensions must be processed; system extensions (those which the *sys. extensions* bit of the primary header set to 1b must be processed by the HPRP engine. The processing of user extensions (where the *user extensions* bit is set to 1b) is implementation-specific and out-of-scope of this document.

Extensions received during reliable sessions must be acknowledged to prevent spurious retransmissions of these extensions, subject to §4.2.7. This procedure shall, for each acknowledgeable extension received within a given data segment, acknowledge them as an entity within the *repeating section* of a metadata acknowledgement extension (§4.2.8.4).

Note: Whether metadata acknowledgments should be aggregated, sent in the next segment, or sent in a stand-alone extension container is a implementation matter.

If the number of pending extensions exceeds the capacity of a single segment extension, multiple metadata acknowledgements may be transmitted.

5.3.4 PROCESS SESSION MANAGEMENT EXTENSIONS

HPRP utilizes session management extensions to signal the status of sessions. Many of these are transmitted from the sender in response to state transitions and failure states. This procedure shall process the extensions.

If a session management extension is received with *Suspension Requested* reason (code = 0x06), all session-specific timers shall be paused for that session. These timers should only be resumed after reception of the next data/extension segment.

Session management extensions which may be transmitted and indicate termination of a session are the following:

- User Cancelled (0x01)
- System Error (0x02)
- Retransmission time exceeded (0x04)
- Retransmission limit exceeded (0x05)
- Session Completed (0x07)

In the case that a failure-related extension is received (0x01-0x05), the session shall terminate and the receiving entity shall indicate failure.

5.3.5 TRANSMIT DATA ACKNOWLEDGEMENT

Upon reception of a data acknowledgement request extension (4.2.8.1), a data acknowledgement (4.2.8.2) must be sent as soon as possible. This procedure shall collect all unacknowledged spans of data (e.g. those which are > the lower bound of successfully-acknowledged data) and transmit them.

If the number of acknowledged regions exceeds the total capacity of an extension header, multiple segments may be sent.

5.3.6 WAIT FOR ACKNOWLEDGEMENT

Waits for acknowledgement of received data. If the lower bound has been expanded (due to successful retransmission), it must not be updated until acknowledgement of the generated data acknowledgement.

5.3.7 NOTIFY SESSION FAILURE

Notify the user of session failure; for reliable sessions, the sender shall be notified of session failures via the transmission of a session management extension.

5.3.8 PROVIDE DATA TO END USER

At the end of a session, resources must be released and data shall be provided to the user (optional if rangeReceived indication is used). If the session has ended in a failure, the receiving implementation may provide the potentially-incomplete data to the user together with the completion map.

6 MANAGED PARAMETERS

The HPRP protocol is intended to satisfy a wide-range of use-cases without requiring SDNV values or other complicated processing tricks. However, some of these use-cases imply differing requirements on field lengths, etc. As a result, multiple length fields exist throughout the protocol. Depending upon mission requirements and the specific field, the lengths of these fields may be:

a. Statically Defined and will not change over an entire mission.

b. Provided within a MIB, static per session/mission phase.

c. Fully-dynamic, using mission-specific heuristics to define behaviour.

Many use-cases (such as high-performance FPGA/ASIC implementations) may not require dynamic reconfiguration of these lengths, allowing the length fields to be provided as parameters during the code generation process. It is also foreseen that on-board implementations will not use dynamic parameters, especially for transmitted data. However, ground-based implementations must allow some degree of variability to support multi-mission operations. The exact source, type (static, MIB, or dynamic) and range of these length fields is mission-specific and should be decided within the relevant tailoring.

Parameter	**Range/unit**	**Used in**	**Description**
Engine ID	1-8 bytes	Protocol Headers	Used to uniquely describe the HPRP entity which is transmitting/receiving. Maximum length defined by the "session originator field length"
Client Service ID	1-8 bytes	Protocol Headers	Describes the destination client service ID.
Acknowledgement Transmission Interval	Bytes and/or duration	Retransmission	The interval at which to transmit data acknowledgement, described in bytes or an implementation-specific duration. If omitted, acknowledgement requests will be only sent at the end of a session. *Note: if durations are used, the unit should be mission-specific. Extremely high-rate applications (such as optical) may require extremely short acknowledgement intervals.*

Metadata Retransmission Limit	Counter	Retransmission	Describes the number of times which a single metadata extension for a given session can be retransmitted before a failure occurs.
Data Retransmission Limit	Counter	Retransmission	Describes the number of times which data can be retransmitted before a failure occurs. *Note: the granularity of this option is implementation-specific; it may either represent the number of retransmissions of a single segment or the total number of retransmissions.*
Metadata Retransmission interval	Seconds	Retransmission	Describes the interval between metadata retransmissions within a given session.
Data Retransmission interval	Seconds	Retransmission	Describes the interval between data retransmissions within a given session.

6.1 DYNAMIC LENGTHS

Table 55 HPRP Dynamic Lengths

Parameter	Range/unit	Type	Description
Session Number Field Length	1-8 bytes	Static	Describes the maximum length of the session number.
Serial Number Field Length	1-8 bytes	Static	Describes the maximum length of an extension serial number.
Session Originator Field Length	1-8 bytes	Static or dynamic	Describes the length of the engine ID in bytes; may be sourced from the “engine ID” field or statically provided.
Client service ID length	1-8 bytes	Static or dynamic	Describes the length of the client service ID in bytes; may be sourced from the “client service ID” field or statically provided.

Data Descriptors Length	1-8 bytes	Static or dynamic	Describes the length of the data descriptors, ie data offset and length information, sourced from the maximum length of the current session or statically provided

It’s foreseen that a high-performance implementation of HPRP may force specific lengths for all potentially dynamic fields. However, the receiving entity may have to support a range of incoming lengths. Two different approaches exist to facilitate this capability: first, the field lengths may be managed in advance, or may be dynamically determined by the protocol. The HPRP protocol signals all length fields within the protocol headers, allowing the dynamic reconfiguration of incoming data and/or the validation of incoming data against the pre-defined “profile” sent as managed parameters

ANNEX A PROTOTYPING

ANNEX B SECURITY CONSIDERATIONS

B1.1 SECURITY CONCERNS WITH RESPECT TO THE CCSDS DOCUMENT

B1.1.1 Data Privacy

B1.1.2 Data Integrity

B1.1.3 Authentication of Communicating Entities

B1.1.4 Control of Access to Resources

B1.1.5 Availability of Resources

B1.1.6 Auditing of Resource Usage

B1.2 POTENTIAL THREATS AND ATTACK SCENARIOS

B1.3 CONSEQUENCES OF NOT APPLYING SECURITY TO THE TECHNOLOGY

ANNEX C SANA CONSIDERATIONS

[See CCSDS 313.0-Y-1, *Space Assigned Numbers Authority (SANA)—Role, Responsibilities, Policies, and Procedures* (Yellow Book, Issue 1, July 2011).]

ANNEX D PATENT CONSIDERATIONS

[See CCSDS A20.0-Y-4, *CCSDS Publications Manual* (Yellow Book, Issue 4, April 2014).]