

**Recommendation for Space Data System Standards**

**CROSS SUPPORT**

**TRANSFER SERVICE—**

**CESG**

**APPROVAL**

**COPY**

**FORWARD FRAME**

**SERVICE**

**DRAFT RECOMMENDED STANDARD**

**CCSDS 922.3-B-0**

# DRAFT BLUE BOOK

**December 2020**

**AUTHORITY**

|  |  |  |
| --- | --- | --- |
| Issue: | Draft Recommended  Issue 0 | Standard, |
| Date: | December 2020 |  |
| Location: | Washington, DC, USA |  |

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 e-mail address below.

This document is published and maintained by:

CCSDS Secretariat

National Aeronautics and Space Administration

Washington, DC, USA

Email: secretariat@mailman.ccsds.org

**STATEMENT OF INTENT**

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. 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 **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

* Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
* Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:

-- The **standard** itself.

-- The anticipated date of initial operational capability.

-- The anticipated duration of operational service.

* Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** 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 canceled.

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

**FOREWORD**

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *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 sent to the CCSDS Secretariat at the email 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 Science Policy Office (BELSPO)/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.
* China 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.
* Electronics and Telecommunications Research Institute (ETRI)/Korea.
* 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.
* Hellenic Space Agency (HSA)/Greece.
* Indian Space Research Organization (ISRO)/India.
* Institute of Space Research (IKI)/Russian Federation.
* Korea Aerospace Research Institute (KARI)/Korea.
* Ministry of Communications (MOC)/Israel.
* Mohammed Bin Rashid Space Centre (MBRSC)/United Arab Emirates.
* 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.
* Netherlands Space Office (NSO)/The Netherlands.
* Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
* 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.

**DOCUMENT CONTROL**

## Document Title Date Status

CCSDS Cross Support Transfer Service— December Original issue

922.3-B-0 Forward Frame Service, Draft 2020

Recommended Standard, Issue 0

**CONTENTS**

### Section

1. **INTRODUCTION** .......................................................................................................... **1-1**
   1. PURPOSE OF THIS RECOMMENDED STANDARD ........................................ 1-1
   2. SCOPE .................................................................................................................... 1-2
   3. APPLICABILITY ................................................................................................... 1-2
   4. RATIONALE .......................................................................................................... 1-3
   5. DOCUMENT STRUCTURE ................................................................................. 1-4
   6. DEFINITIONS ........................................................................................................ 1-8
   7. NOMENCLATURE ............................................................................................. 1-12
   8. CONVENTIONS .................................................................................................. 1-13
   9. REFERENCES ..................................................................................................... 1-13

## 2 OVERVIEW OF THE FORWARD FRAME CROSS SUPPORT

**TRANSFER SERVICE** ................................................................................................. **2-1**

2.1 SERVICE SUMMARY .......................................................................................... 2-1

2.2 FUNCTIONAL DESCRIPTION ............................................................................ 2-2

2.3 SERVICE MANAGEMENT ................................................................................ 2-11

2.4 CROSS SUPPORT VIEW .................................................................................... 2-13

2.5 OPERATIONAL SCENARIOS ........................................................................... 2-15

## 3 COMPOSITION OF THE FORWARD FRAME CROSS SUPPORT

**TRANSFER SERVICE** ................................................................................................. **3-1**

3.1 DISCUSSION ......................................................................................................... 3-1

3.2 DATA PROCESSING MODE SERVICE MANAGEMENT PARAMETER ...... 3-1

3.3 PROCEDURES OF THE SEQUENCE-CONTROLLED DATA PROCESSING MODE OF THE FORWARD FRAME CROSS SUPPORT

TRANSFER SERVICE .......................................................................................... 3-2

3.4 PROCEDURES OF THE BUFFERED DATA PROCESSING MODE

OF THE FORWARD FRAME CROSS SUPPORT TRANSFER SERVICE ........ 3-6 3.5 FORWARD FRAME CROSS SUPPORT TRANSFER SERVICE

STATE MACHINE ................................................................................................ 3-9

## 4 SEQUENCE-CONTROLLED FRAME DATA PROCESSING PROCEDURE ..... 4-1

4.1 DISCUSSION ......................................................................................................... 4-1

4.2 PROCEDURE TYPE IDENTIFIER ....................................................................... 4-2

4.3 REFINEMENT ....................................................................................................... 4-2

4.4 EXTENSION .......................................................................................................... 4-2 4.5 BEHAVIOR ............................................................................................................ 4-3

**CONTENTS (continued)**

### Section Page

4.6 REQUIRED OPERATIONS .................................................................................. 4-5

4.7 CONFIGURATION PARAMETERS .................................................................... 4-8

4.8 READ-ONLY PARAMETERS ............................................................................ 4-10

4.9 PROCEDURE STATE TABLE ........................................................................... 4-11

## 5 BUFFERED FRAME DATA PROCESSING PROCEDURE ................................... 5-1

5.1 DISCUSSION ......................................................................................................... 5-1

5.2 PROCEDURE TYPE IDENTIFIER ....................................................................... 5-2

5.3 REFINEMENT ....................................................................................................... 5-2

5.4 EXTENSION .......................................................................................................... 5-2

5.5 BEHAVIOR ............................................................................................................ 5-3

5.6 REQUIRED OPERATIONS .................................................................................. 5-5

5.7 CONFIGURATION PARAMETERS .................................................................... 5-9

5.8 READ-ONLY PARAMETERS ............................................................................ 5-11

5.9 PROCEDURE STATE TABLE ........................................................................... 5-13

1. **MASTER THROW EVENT PROCEDURE** ............................................................... **6-1**

* 1. DISCUSSION ......................................................................................................... 6-1
  2. PROCEDURE TYPE IDENTIFIER ....................................................................... 6-2
  3. REFINEMENT ....................................................................................................... 6-2
  4. EXTENSION .......................................................................................................... 6-2
  5. BEHAVIOR ............................................................................................................ 6-2
  6. REQUIRED OPERATIONS .................................................................................. 6-3
  7. CONFIGURATION PARAMETERS .................................................................... 6-4
  8. PROCEDURE STATE TABLE ............................................................................. 6-4

## 7 SETTING OF SERVICE MANAGEMENT AND CONFIGURATION PARAMETERS INHERITED FROM FRAMEWORK OPERATIONS

**AND PROCEDURES** .................................................................................................... **7-1**

7.1 GENERAL .............................................................................................................. 7-1

7.2 ASSOCIATION CONTROL PROCEDURE CONFIGURATION

PARAMETERS ...................................................................................................... 7-1

7.3 SEQUENCE-CONTROLLED FRAME DATA PROCESSING

PROCEDURE CONFIGURATION PARAMETERS ........................................... 7-1

7.4 BUFFERED FRAME DATA PROCESSING PROCEDURE

CONFIGURATION PARAMETERS .................................................................... 7-2

7.5 CYCLIC REPORT PROCEDURE CONFIGURATION PARAMETERS............ 7-2

**CONTENTS (continued)**

### Section

7.6 NOTIFICATION PROCEDURE CONFIGURATION PARAMETERS .............. 7-3 7.7 INFORMATION QUERY PROCEDURE CONFIGURATION

PARAMETERS ...................................................................................................... 7-3

7.8 MASTER THROW EVENT PROCEDURE CONFIGURATION

PARAMETERS ...................................................................................................... 7-3

## 8 FORWARD FRAME SERVICE-LEVEL FUNCTIONAL RESOURCE

**PARAMETERS AND EVENTS** ................................................................................... **8-1**

8.1 GENERAL .............................................................................................................. 8-1

8.2 FFRESPONSETIMEOUT FUNCTIONAL RESOURCE PARAMETER .............. 8-1

8.3 FFPRODSTAT FUNCTIONAL RESOURCE PARAMETER ............................... 8-2

8.4 FFPRODSTATCHANGE EVENT ............................................................................ 8-2

8.5 FFPRODCONFIGURATIONCHANGE EVENT ...................................................... 8-3

## 9 REFINEMENT OF DEFINITIONS OF FRAMEWORK PARAMETERS, EVENTS, DIRECTIVES, AND DIAGNOSTIC VALUES USED BY THE

**FORWARD FRAME SERVICE** .................................................................................. **9-1**

9.1 GENERAL .............................................................................................................. 9-1

9.2 ‘UNSUPPORTED OPTION’ DIAGNOSTIC VALUE REFINEMENT ................ 9-1

**ANNEX A IMPLEMENTATION CONFORMANCE STATEMENT**

**PROFORMA (NORMATIVE)** .................................................................... **A-1**

**ANNEX B SERVICE OBJECT IDENTIFIERS (NORMATIVE)** ..............................**B-1** **ANNEX C PROCEDURE—SEQUENCE-CONTROLLED FRAME**

**DATA PROCESSING PDUS (NORMATIVE)** .......................................... **C-1**

**ANNEX D PROCEDURE—BUFFERED FRAME DATA PROCESSING**

**PDUS (NORMATIVE)** ................................................................................. **D-1**

**ANNEX E PROCEDURE—MASTER THROW EVENT PDUS**

**(NORMATIVE)** ..............................................................................................**E-1**

**ANNEX F FORWARD FRAME SERVICE PROCEDURE PARAMETERS,**

**EVENTS, AND DIRECTIVES (NORMATIVE)** ........................................ **F-1**

**ANNEX G FORWARD FRAME SERVICE PRODUCTION**

**REQUIREMENTS (NORMATIVE)** ........................................................... **G-1**

**ANNEX H SECURITY, SANA, AND PATENT CONSIDERATIONS**

**(INFORMATIVE)** ......................................................................................... **H-1**

## ANNEX I EXAMPLE FORWARD FRAME PROVISION AND PRODUCTION CONFIGURATIONS (INFORMATIVE) ........................ I-1 ANNEX J EXAMPLE EXTENDED USE CASES FOR THE FORWARD

**FRAME SERVICE (INFORMATIVE)** ....................................................... **J-1**

**CONTENTS (continued)**

Section Page

## ANNEX K USING THE FORWARD FRAME SERVICE TO SHARE ACCESS TO A FORWARD LINK WITH DTN BUNDLES (INFORMATIVE) ... K-1 ANNEX L INFORMATIVE REFERENCES (INFORMATIVE) ...............................L-1 ANNEX M ABBREVIATIONS AND ACRONYMS (INFORMATIVE) ................... M-1 ANNEX N CROSS REFERENCES TO CROSS SUPPORT TRANSFER SERVICE SPECIFICATION FRAMEWORK (INFORMATIVE) ........ N-1

### Figure

1-1Cross Support Services Documentation ....................................................................... 1-6

2-1Production and Provision of Forward Frame Cross Support Transfer Service—Transfer

Frame Configuration ..................................................................................................... 2-2

2-2Production and Provision of Forward Frame Cross Support Transfer Service—CADU

Configuration ................................................................................................................ 2-2

2-3Cross Support View of Forward Frame Services—Transfer Frame Configuration ... 2-14

2-4Cross Support View of Forward Frame Services—CADU Configuration................. 2-14

2-5Forward Frame Service Scenario—Transfer Frame Configuration—TC Frames...... 2-15

2-6Forward Frame Service Scenario—Transfer Frame Configuration—AOS Frames ... 2-22

2-7Forward Frame Service Scenario—CADU Configuration ......................................... 2-26

I-1Forward Frame Service—Transfer Frame Configuration—TC Frames ........................ I-3

I-2Forward Frame Service—Transfer Frame Configuration—AOS Frames ..................... I-7I-3Forward Frame Service—Transfer Frame Configuration—Variable-Length USLP

Frames .......................................................................................................................... I-12

I-4Forward Frame Service—Transfer Frame Configuration—Fixed-Length USLP FramesI-16

I-5Forward Frame Service—CADU Configuration ......................................................... I-19J-1Forward Frame Service—Extended CADU Configuration—SMTFs ........................... J-4

K-1DTN ABA Architecture ............................................................................................... K-2

K-2DTN Relay Satellite Architecture ................................................................................ K-3

K-3DTN and Forward Frame over Telecommand SDLP .................................................. K-5

### Table

3-1Procedures of the Sequence-Controlled Data Processing Mode of the Forward Frame

Service .......................................................................................................................... 3-4

3-2Procedures of the Buffered Data Processing Mode of the Forward Frame Service ..... 3-8

4-1Sequence-Controlled Frame Data Processing Procedure Required Operations ........... 4-6

4-2START Extension Parameter........................................................................................ 4-6

4-3Sequence-Controlled Frame Data Processing Procedure Configuration Parameters ... 4-9

4-4Sequence-Controlled Frame Data Processing Procedure Read-Only Parameters ...... 4-10

4-5Sequence-Controlled Frame Data Processing Procedure State Table ........................ 4-11

**CONTENTS (continued)**

### Table

4-6Procedure State Table Incoming Event Description References ................................ 4-13

4-7Procedure State Table Predicate Descriptions ............................................................ 4-134-8Procedure State Table Simple Action References ...................................................... 4-144-9Procedure State Table Compound Action Definitions ............................................... 4-15

5-1Buffered Frame Data Processing Procedure Required Operations ............................... 5-6

5-2START Extension Parameter........................................................................................ 5-6

5-3Buffered Frame Data Processing Procedure Configuration Parameters ..................... 5-10

5-4Buffered Frame Data Processing Procedure Read-Only Parameters ......................... 5-12

5-5Buffered Frame Data Processing Procedure State Table ............................................ 5-135-8Procedure State Table Boolean Flags ......................................................................... 5-165-10Procedure State Table Compound Action Definitions ............................................... 5-17

6-1Master Throw Event Procedure Required Operation ................................................... 6-3

6-2Master Throw Event Procedure Configuration Parameters .......................................... 6-4

A-1Identification of PICS .................................................................................................. A-4

A-2Identification of Implementation Under Test .............................................................. A-4

A-3Identification of Supplier ............................................................................................. A-4

A-4Identification of Specification ..................................................................................... A-4

A-5Procedures Used by the FF-CSTS Specification ......................................................... A-5

A-6Required PDUs ............................................................................................................ A-6

A-7BIND Invocation Parameters ....................................................................................... A-8

A-8BIND Return Parameters ............................................................................................. A-9

A-9PEER-ABORT Invocation Parameters ...................................................................... A-10

A-10UNBIND Invocation Parameters ............................................................................... A-10

A-11UNBIND Return Parameters ..................................................................................... A-11

A-12EXECUTE-DIRECTIVE Invocation Parameters ...................................................... A-11

A-13EXECUTE-DIRECTIVE Acknowledgement Parameters ......................................... A-13

A-14EXECUTE-DIRECTIVE Return Parameters ............................................................ A-14

A-15GET Invocation Parameters ....................................................................................... A-16

A-16GET Return Parameters ............................................................................................. A-16

A-17PROCESS-DATA Invocation Parameters ................................................................. A-18

A-18PROCESS-DATA Return Parameters ....................................................................... A-20

A-19START Invocation Parameters .................................................................................. A-22

A-20START Return Parameters ........................................................................................ A-24A-21STOP Invocation Parameters ..................................................................................... A-26A-22STOP Return Parameters ........................................................................................... A-27

A-23NOTIFY Invocation Parameters ................................................................................ A-28

A-24TRANSFER-DATA Invocation Parameters .............................................................. A-30

N-1Cross References to CSTS Specification Framework Subsections ............................. N-1

## 1 INTRODUCTION

### 1.1 PURPOSE OF THIS RECOMMENDED STANDARD

This Recommended Standard defines the Forward Frame Cross Support Transfer Service (FFCSTS). The FF-CSTS is a transfer service that provides the transfer of fixed-length and variablelength Transfer Frames and Channel Access Data Units (CADUs) to a Provider Cross Support Service System (CSSS) Earth-Space Link Terminal (ESLT, e.g., a ground station) to be encoded (as needed) and radiated to a Space User Node (e.g., a Mission spacecraft). The supported variable-length Transfer Frames include Telecommand (TC) Transfer Frames defined in reference [L12] and variable-length Unified Space Data Link (USLP) Transfer Frames defined in reference [L14]. The supported fixed-length Transfer Frames include the Advanced Orbiting System (AOS) Transfer Frames defined in reference [L13] and fixed-length USLP Transfer Frames defined in reference [L14]. The supported CADU format is defined in reference [L15].

NOTES

1. The CADU defined in reference [L15] is the CCSDS-standard data unit that carries fixed-length transfer frames into the physical channel and is therefore applicable to AOS Transfer Frames and fixed-length USLP Transfer Frames. As of the publication of this Recommended Standard, the CADU as defined in the current version of reference [L15] applies only in the context of telemetry data transmitted on a space-to-ground link. CCSDS is in the process of revising reference [L15] to specify which of the coding and synchronization functions and formats that are defined therein will be applicable to the transmission of fixed-length Transfer Frames on the forward link, and to make the CADU a data unit of the forward link.
2. Although the Forward Frame service is defined to support the use of TC, AOS, and USLP Transfer Frames and CADUs on the forward link, the transfer service may be used to handle other space data link and/or channel synchronization data unit formats. The use of the Forward Frame service to support such other data unit formats is outside the normative scope of this Recommended Standard. However, informative annex J presents examples of use cases in which the transfer service can be used beyond the configurations defined in this Recommended Standard.
3. In this Recommended Standard, the term *Space Link Protocol Data Unit* (SL-PDU) refers to either a Transfer Frame or a CADU. It is used in statements that apply to both Transfer Frames and CADUs.

When used to carry Transfer Frames, multiple instances of the Forward Frame service can exist for the same single forward space link physical channel, and each service instance carries the frames from one Virtual Channel (VC) (see references [L12]–[L14]). Conversely, when used to carry CADUs, the forward space link physical channel is dedicated to a single instance of the Forward Frame service.

When Forward Frame service instances are used to carry individual VCs, those VCs may be multiplexed with VCs from other non-Forward Frame service sources. Such multiplexing is not a function of the Forward Frame service but is a function of the underlying production processing.

### 1.2 SCOPE

This Recommended Standard defines the FF-CSTS using procedures and operations defined in the *Cross Support Transfer Service Specification Framework* (reference [1]) and in accordance with the *Guidelines for the Specifications of Cross Support Transfer Services* (reference [3]).

Specifically, the Recommended Standard is defined in terms of

1. the CSTS Specification Framework (SFW) procedures that comprise the Forward Frame service. The SFW procedures used in any implementation of this Issue (Issue-1) of this Recommended Standard must conform to the selected procedures as specified in Issue-2 of the SFW as specified in reference [1]. Any future Issues of this Recommended Standard (if any) will reference and conform to the Issue of the SFW that is in effect at that time;
2. the extensions and refinements of the behavior of those CSTS procedures necessary to provide the transfer service;
3. the extensions and refinements of standard CSTS operations associated with each of the procedures;
4. the relationships among the procedures that comprise the service; and
5. the requirements on Forward Frame service production to enable the proper operation of the FF-CSTS.

This Recommended Standard does not specify

1. individual implementations or products;
2. the implementation of entities or interfaces within real systems;
3. the methods or technologies required to measure the values of Forward Frame service parameters;
4. the methods or technologies required for communication; or
5. the management activities necessary to schedule, configure, and control the FF-CSTS.

### 1.3 APPLICABILITY

The applicability and limits of applicability of CSTSes in general, as described in [1], pertain to the FF-CSTS.

The FF-CSTS is applicable in situations in which there is a need for multiple users to concurrently send data destined for the same space link physical channel to a Space User Node. Each Forward Frame service instance delivers the Transfer Frames from its user to the appropriate underlying production multiplexing, synchronization and channel coding functions (as defined in references [L12], [L13], [L14], and in a planned revised version of reference [L15], as applicable).

The FF-CSTS is also applicable in situations in which the Earth User Node performs the multiplexing and coding of all fixed length transfer frames to produce all of the user-databearing CADUs destined for the same forward space link physical channel.

The Forward Frame service operates in one of *two data processing modes*, the *sequencecontrolled data processing mode* (see 1.6.1.12.2.1) and the *buffered data processing mode* (see 1.6.1.12.2.2), which allows the service user to trade off between frame-by-frame control of transmission vs. data throughput. Each instance of the Forward Frame service is configured to be in a single data processing mode for the duration of the execution of a Service Package.

The FF-CSTS enables the encoding, randomization, synchronization marking, and Only Idle Data (frames or CADUs, as appropriate) insertion functions to be performed by the production processes of the ESLT. Performing these functions in the ESLT instead of in the Earth User Node improves the performance of the communications link, creating a more robust forward space link service to users. These encoding, randomization, synchronization marking, and Only Idle Data insertion functions are not part of the Forward Frame transfer service; they are configured by Service Management function to work cooperatively with the Forward Frame transfer service instance(s) to collectively process the user-supplied transfer frames or CADUs and transmit them to the User Space Node.

### 1.4 RATIONALE

The rationale for this Recommended Standard is

1. to create a standard for interoperability for the exchange of fixed-length transfer frames, variable-length transfer frames, and CADUs between the cross-support elements of various space agencies and the users of the cross-support services provided by these agencies;
2. to permit Earth User Node sources of VC transfer frames to share the same forward space link physical channel with other VC transfer frame sources, including, but not limited to, other Forward Frame service instances, Delay Tolerant Network (DTN) bundle agents (see annex K), and Internet Protocol (IP) routers; and
3. to permit Earth User Nodes to generate and transfer CADUs to be transmitted on a forward space link physical channel.

Four use cases motivate the existence of the Forward Frame service:

1. The ability to transfer TC Transfer Frames (see reference [L12]) and variable-length USLP Transfer Frames Transfer Frames (see reference [L14]) with sequence and timing enforcement behavior comparable to that provided for CLTUs by the SLE Forward CLTU service (reference [L4]) and for TC Space Packets by the SLE Forward Space Packet service (reference [L19]); this use case is illustrated in annexes I2 and I4.
2. The ability to transfer AOS Transfer Frames (see reference [L13]) and USLP fixedlength Transfer Frames (see reference [L14]) with maximum throughput and with no sequence or timing enforcement (by the transfer service itself). This use case is illustrated in annexes I3 and I5.
3. The ability to transfer fixed-length CADUs (see reference [L15]) with maximum throughput and with no sequence or timing enforcement (by the transfer service itself). This use case provides the fixed-length frame equivalent of the SLE Forward CLTU service (reference [L4]). This use case is illustrated in annex I6.
4. The ability to multiplex data from different sources into a single space link physical channel or master channel including specifically sources that accept DTN bundles or IP datagrams, and process them to create correctly formatted transfer frames for multiplexing. This use case provides services critical to cross supported Solar System Internetworking (SSI) networking configurations [L21]. This use case is illustrated in annex K.

However, the Forward Frame service is not restricted to these three uses cases, and other combinations are possible. For example, variable-length transfer frames could be transferred by a Forward Frame service instance that is configured to operate in the data processing mode that maximizes throughput and performs no sequencing or timing enforcement. Annex J provides several examples of such extended use cases.

### 1.5 DOCUMENT STRUCTURE

#### 1.5.1 DOCUMENT ORGANIZATION

Section 2 describes the FF-CSTS in terms of

* the functional description of the production and provision of the service;
* the role of Service Management with respect to the FF-CSTS;
* the cross-support view of the FF-CSTS; and
* operational scenarios that illustrate some of the more significant aspects of the service.

Section 3 specifies the top-level composition of the FF-CSTS. The service-type identifier is declared, the procedures that comprise the service are identified, and the CSTS state machine that applies to the FF-CSTS is specified.

Section 4 defines the Sequence Controlled Frame Data Processing (SCFDP) procedure as a derivation of the CSTS SFW Sequence Controlled Data Processing procedure.

Section 5 defines the Buffered Frame Data Processing (BFDP) procedure as a derivation of the CSTS SFW Buffered Data Processing procedure.

Section 6 defines the refinements to the CSTS SFW Throw Event procedure as it is used by the Forward Frame service.

Section 7 specifies how the procedure configuration parameters are to be set for the FF-CSTS.

Section 8 specifies the Forward Frame Service-specific versions of the service-level parameters and events that are required by the CSTS SFW to be present in all CSTSes.

Section 9 defines the refinement of CSTS SFW-defined parameters and events as they apply to the FF-CSTS.

Annex A is the Implementation Conformance Statement (ICS) Proforma for the Forward Frame service.

Annex B is the Abstract Syntax Notation One (ASN.1) module that formally specifies the object identifiers for the Forward Frame transfer service and the FF-CSTS Provider Functional Resource (FR) Type.

Annex C formally specifies the ASN.1 PDUs for the Sequence Controlled Frame Data Processing procedure.

Annex D formally specifies the ASN.1 PDUs for the BFDP procedure.

Annex E formally specifies the ASN.1 PDUs for the Master Throw Event procedure.

Annex F formally specifies the ASN.1 parameters, events, and directives for the Sequence Controlled Frame Data Processing and BFDP procedures.

Annex G identifies the space data link protocol, synchronization and channel coding, space data link physical channel, and aperture functions that are performed by the production process that underlies the Forward Frame service and defines the Forward Frame-specific requirements on that production process.

Annex H addresses the security, Space Assigned Numbers Authority (SANA), and patent considerations associated with the FF-CSTS.

Annex I provides examples of the use of the Forward Frame service and the associated production functions to support the transmission of TC frames, AOS frames, variable-length USLP frames, fixed-length frames, and CADUs.

Annex J provides examples of use cases in which the Forward Frame service can be used outside of the use cases for which the service is defined.

Annex K describes how the Forward Frame service enables an ESLT to receive DTN bundles and Mission-generated transfer frames and multiplex them into a single forward space link.

Annex L lists the informative references cited in this Recommended Standard.

Annex M lists the abbreviations and acronyms used in this Recommended Standard.

Annex N lists the cross-references to the CSTS SFW (reference [1]).

#### 1.5.2 CROSS SUPPORT TRANSFER SERVICES DOCUMENTATION

The basic organization of the Cross Support Services documentation and the relationship to the CSTS documentation is shown in figure 1-1.

|  |
| --- |
| ***Cross Support Documents*** |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | ***Cross Support Architecture*** | | |  | | --- | | Space Communication  Cross Support –  Architecture Description Document | | Space Communication  Cross Support –  Architecture  Requirements Document | | | |  | | --- | | ***Cross Support Services*** | | |  | | --- | | ***Transfer Services*** | | |  |  |  |  | | --- | --- | --- | --- | | ***Cross Support Transfer Services*** | | | | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Cross Support Transfer Service Concept |  | | Guidelines for  Specification of Cross  Support Transfer Services | | | |  | | --- | | Cross Support Transfer  Service Specification Framework | | | |  |  |  | | --- | --- | --- | |  |  |  | | Cross Support Transfer  Service Specification Suite |  | |  |  | | | | | | | | | ***SLE***    ***Transfer***    ***Services*** | | | | |  |  |  | | --- | --- | --- | | Cross Support Concept Part 1: SLE Services |  | Cross Support Reference  Model  Part 1: SLE Services |   SLE    Transfer    Services    Specifications    Suite | | | | SLE Internet Protocol for Transfer Services | |   Space Communication  Cross Support Service Management Suite | | |

|  |  |  |
| --- | --- | --- |
| **Recommended Standard (Blue)** |  | **Recommended Practice (Magenta)** |

Key:

**Report (Green)**

#### Figure 1-1: Cross Support Services Documentation

The Cross Support Architecture is documented in

1. *Space Communications Cross Support—Architecture Description Document* (reference [L8]): An Informational Report describing an architecture in terms of CCSDS-recommended configurations for secure space communications cross support. This architecture is intended to be used as a common framework when CCSDS Agencies a) provide and use space communications cross support services and b) develop systems that provide interoperable space communications cross support.
2. *Space Communications Cross Support—Architecture Requirements Document* (reference [L21]): A Recommended Practice defining a set of requirements for CCSDSrecommended configurations for secure space communications cross support architectures.

Common to all Cross Support Services

1. *Space Communication Cross Support Service Management suite* (references [L5], [L6], and [L7]). Data format Recommended Standards that specify the Service Management Information Entities that are used to configure and schedule cross support services, which include transfer services.

Common to the Transfer Services, that is, SLE Transfer Services and CSTSes

1. *Space Link Extension—Internet Protocol for Transfer Services* (reference [L2]): A Recommended Standard that defines a protocol for transfer of Protocol Data Units (PDUs) defined in the CSTSes. This Recommended Standard was originally developed to support SLE transfer services (hence the title), but it is also applicable to CSTSes.

The concept, the reference model, and the SLE Transfer Services are documented in

1. *Cross Support Concept—Part 1: Space Link Extension Services* (reference [L1]): A report introducing the concepts of cross support and the SLE services. Many of the concepts for the SLE transfer services have been adopted for the CSTSes (see k) below).
2. *Cross Support Reference Model—Part 1: Space Link Extension Services* (reference [2]): A Recommended Standard that defines the framework and terminology for the specification of SLE services. Much of the framework and terminology of this reference model has been adopted or adapted for CSTSes (see 1.6.3.1 and 2.2 of reference [1]).
3. The *SLE Transfer Services suite*: The SLE Transfer Services are a suite of Cross Support Services that are used to transfer specific TC and TM PDUs. The SLE Transfer Services are closely related to the CSTS suite in that they collectively define the set of operations that are the basis for the CSTS SFW. However, because of their history (the SLE Transfer Services were already specified and implemented prior to development of the CSTS SFW), the SLE Transfer Services are separated from CSTSes.

The documents specific to CSTSes are

1. *Cross Support Transfer Services Specification Framework* (reference [1]): A Recommended Standard that defines the specification of the CSTS procedures;
2. *Guidelines for Specification of Cross Support Transfer Services* (reference [3]): A Recommended Practice that defines the guidelines for construction of a CSTS based on the CSTS SFW;
3. *Cross Support Transfer Services Concept* (reference [L3]): A Report that provides tutorial material on the objectives and concepts of the CSTS specification; and
4. *Cross Support Transfer Services Suite*: The set of specifications for actual CSTSes built from the procedures in the CSTS SFW and in accordance with the CSTS Guidelines. This Recommended Standard is a member of this suite.

### 1.6 DEFINITIONS

#### 1.6.1 TERMS

##### 1.6.1.1 Terms Defined in the Cross Support Transfer Services Specification Framework, Reference [1]

1. Association Control procedure;
2. blocking [operation];
3. Buffered Data Processing procedure;
4. complete [transfer mode];
5. CSTS;
6. functional resource;
7. latency-limit;
8. procedure configuration parameter;
9. ProcessDataInvocation;
10. qualified parameter;
11. Sequence Controlled Data Processing procedure;
12. service instance provision period;
13. service management parameter;
14. timely [transfer mode].

##### 1.6.1.2 Terms Defined in the Cross Support Reference Model, Reference [2]

1. Mission User Entity (MUE);
2. Service Package;
3. space link session.

##### 1.6.1.3 Terms Defined in the Space Communications Cross Support Architecture Description Document, Reference [L8]

1. CSSS;
2. ESLT;
3. Earth User CSSS;
4. Earth User Node;
5. Element Management (EM);
6. Provider CSSS;
7. Provision Management (PM);
8. Service Management;
9. Space User Node;
10. Utilization Management (UM).

##### 1.6.1.4 Terms Defined in Abstract Syntax Notation One, Reference [5]

1. ASN.1;
2. Object Identifier.

NOTE – In annexes B, C, D, E, and F of this Recommended Standard, ASN.1 is used for specifying the Object Identifiers and the abstract syntax of the PDUs, parameters, events, and directives of the Forward Frame service.

##### 1.6.1.5 Terms Defined in the TC, AOS, and Unified Space Data Link Protocols, References [L12], [L13], [L14], Respectively

1. Communications Operation Procedure (COP) (reference [L12] only);
2. Global Virtual Channel Identifier (GVCID);
3. Only Idle Data Frame (references [L13] and [L14] only);
4. Master Channel (MC);
5. Master Channel Identifier (MCID);
6. Spacecraft Identifier (SCID);
7. Transfer Frame (the format is specific to each space data link protocol);
8. Transfer Frame Primary Header;
9. Transfer Frame Version Number (TFVN);
10. VC; and
11. Virtual Channel Identifier (VCID).

##### 1.6.1.6 Terms Defined in the TM Synchronization and Channel Coding Recommended Standard, Reference [L15]

1. Attached Synchronization Marker (ASM);
2. CADU;
3. Low-Density Parity-Check (LDPC);
4. Low-Density Parity-Check coding of a stream of Sync-Marked Transfer Frames;
5. Sync-Marked Transfer Frame (SMTF).

##### 1.6.1.7 Terms Defined in this Recommended Standard

**1.6.1.7.1 bit masking:** As used in this Recommended Standard, bit masking is the method used by Forward Frame services to validate the subfields of the first four octets of the SLPDU that correspond to the TFVN, SCID, and VCID (which collectively constitute the GVCID) of the Transfer Frame Primary Header of the CCSDS TC, AOS, and USLP Transfer Frames. Because these subfields are of different lengths and appear in different locations within the first four octets of each type of Transfer Frame, a different bit mask exists for each Transfer Frame type. Within each Transfer Frame type’s bit mask, the bits of the TFVN, SCID, and VCID subfields are all set to one (1). When a Forward Frame service instance receives an SL-PDU, it ANDs the bit mask with the first four octets of that SL-PDU. If the TFVN, SCID, and VCID subfields of the product of the ANDing are equal to the values of the TFVN, SCID, and VCID subfields of the four-octet *authorized GVCID* configuration parameter for that Forward Frame service instance.

**1.6.1.8** For example, for TC transfer frames, the TFVN appears in bits 0 and 1, the SCID appears in bits 6–15, and the VCID appears in bits 16-21 of the first 4 octets of the Transfer Frame Primary Header. The corresponding GVCID bit mask is **1.6.1.9** 11000011 11111111 11111100 00000000 (C3 FF FC 00 Hex)

**1.6.1.10** where the spaces between the octet values are for readability.

**1.6.1.11** For a Forward Frame service instance configured to process TC transfer frames for the GVCID (TFVN = 0, SCID = 20, VCID = 3), the corresponding authorized GVCID configuration parameter has the value

**1.6.1.12** 00000000 00010100 00001100 00000000 (00 14 0C 00 Hex)

NOTE – Bit masking is intended for use in Transfer Frame configurations, where multiple Forward Frame service instances for the same space link physical channel are possible and therefore each Forward Frame service instance needs to authenticate transfer frames to ensure that they are of the authorized GVCID. In the CADU configuration there is only a single Forward Frame instance for each physical channel, so the bit mask and authorized GVCID can both be set to all zeroes—that is, ‘don’t care’—to allow all CADUs to be authenticated regardless of the content of their first four octets (nominally, all or part of the Attached Synchronization Marker).

**1.6.1.12.1 CADU configuration:** A combination of a single Forward Frame transfer service instance and underlying production functions that are configured by Service Management to support the transfer of CADUs from an Earth User Node to the ESLT performing the Forward Frame service instances. The single Forward Frame service instance provides all of the CADUs for the associated space link physical channel.

**1.6.1.12.2 data processing mode:** The mode in which a Forward Frame service instance is configured (via Service Management) to process received SL-PDUs. A Forward Frame service instance can be in only one data processing mode for the duration of the execution of a Service Package. The data processing mode has one of two values:

**1.6.1.12.2.1 sequence-controlled data processing mode:** The sequence-controlled data processing mode allows the service user to place constraints on each SL-PDU regarding (1) the sequence in which the SL-PDU must be received (with respect to the previous SL-PDU) by the Forward Frame service provider and (2) the time window in which the SL-PDU must be submitted to production processing. (See 2.1a) for details.)

NOTE – The sequence-controlled data processing mode of the Forward Frame service is not to be confused with the Sequence-Controlled Service of the TC and Unified Space Data Link Protocols (SDLPs) (references [L12] and [L14], respectively). If they are performed, the TC and USLP Sequence-Controlled Services are performed by the MUE of the Earth User Node prior to transfer of the Transfer Frames through the Forward Frame service.

**1.6.1.12.2.2 buffered data processing mode:** In order to maximize throughput, the buffered data processing mode does not regulate the flow of SL-PDUs to the underlying production process. Increased throughput is also supported by the capability to place multiple SL-PDUs into a single Forward Buffer PDU in order to maximize the efficiency of the underlying communication service. (See 2.1b) for details.)

**1.6.1.12.3 functional resource parameter**: a functional resource parameter of a CSTS is a parameter that is accessible through the functional resource that represents that CSTS. A functional resource parameter is named using the Functional Resource Type and functional resource parameter Published Identifiers registered in the SANA Functional Resource Registry (reference [4]). A service management parameter (see 1.6.1.1 m)) is a functional resource parameter that is used to configure the CSTS that is represented by its associated CSTS functional resource. In this Recommended Standard, the term ‘service management parameter’ is used for functional resource parameters that are used to configure the CSTS, and the more-general ‘functional resource parameter’ is used for read-only (i.e., non-configuration) parameters of the CSTS.

**1.6.1.12.4 Only Idle Data CADU:** The Only Idle Data CADU is a CADU that is created to preserve the continuity of the transmission stream in the even that there are no valid CADUs available for transmission at release time. Only Idle Data CADUs are generated only when the Forward Frame service is operating in the CADU configuration (see 2.2.2.2 and I6). The generation and injection of Only Idle Data CADUs results in a transmitted waveform that conforms to CCSDS-standard space data link and synchronization and channel coding standard for fixed length frames.

**1.6.1.12.5 Only Idle Data SMTF:** The Only Idle Data SMTF is a SMTF that is created to preserve the continuity of the transmission stream in the even that there are no valid SMTFs available for transmission at release time. Only Idle Data SMTFs are generated only when the Forward Frame service is used to transfer SMTFs for subsequent encoding through LDPC of a stream of SMTFs (see [L15]). Transfer of SMTFs through the Forward Frame service is accomplished through a hybrid of the CADU configuration (see 2.2.2.2 and I6) and LDPC of a stream of SMTFs, as described in J5. The generation and injection of Only Idle Data SMTFs results in a transmitted waveform that conforms to CCSDS-standard space data link and synchronization and channel coding standard for fixed length frames.

**1.6.1.12.6 Space Link Protocol Data Unit, SL-PDU:** an SL-PDU is either a Transfer Frame or a CADU. The term is used in statements that apply to both Transfer Frames and CADUs.

**1.6.1.12.7 Transfer Frame configuration:** A combination of Forward Frame transfer service instances and underlying production functions that are configured by Service Management to support the transfer of Transfer Frames from an Earth User Node to the ESLT performing the Forward Frame service instances. Multiple Forward Frame service instances can provide transfer frames for the same space link physical channel in a Transfer Frame configuration, where each Forward Frame service instance nominally carries the frames for a single Virtual Channel.

### 1.7 NOMENCLATURE

#### 1.7.1 NORMATIVE TEXT

The following conventions apply for the normative specifications in this Recommended Standard:

1. the words ‘shall’ and ‘must’ imply a binding and verifiable specification;
2. the word ‘should’ implies an optional, but desirable, specification;
3. the word ‘may’ implies an optional specification;
4. 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.

#### 1.7.2 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.

**1.8 CONVENTIONS**

This Recommended Standard uses the conventions defined in the CSTS SFW (reference [1]).

### 1.9 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. It is important to note that the edition of the *Cross Support Transfer Service— Specification Framework* that is named in reference [1] is the only edition that can be used as a source of requirements for this edition of this Cross Support Transfer Service Recommended Standard. All other publications in the list are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the publications (other than the Specification Framework) indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS publications.

1. *Cross Support Transfer Service—Specification Framework*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 921.1-B-2. Washington, D.C.: CCSDS, forthcoming.
2. *Cross Support Reference Model—Part 1: Space Link Extension Services*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 910.4-B-2. Washington, D.C.: CCSDS, October 2005.
3. *Guidelines for the Specification of Cross Support Transfer Services*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 921.2-M-1. Washington, D.C.: CCSDS, March 2019.
4. “Functional Resources Registry” Space Assigned Numbers Authority. http://sanaregistry.org/r/functional\_resources/.
5. *Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*. 5th ed. International Standard, ISO/IEC 8824-1:2015. Geneva: ISO, 2015.
6. *Information Technology—ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*. 5th ed. International Standard, ISO/IEC 8825-1:2015. Geneva: ISO, 2015.

## 2 OVERVIEW OF THE FORWARD FRAME CROSS SUPPORT TRANSFER SERVICE

### 2.1 SERVICE SUMMARY

The Forward Frame service is a CCSDS CSTS that provides the user with the capability to transfer SL-PDUs for radiation to a Space User Node (e.g., spacecraft) in one of two data processing modes (as configured via Service Management):

1. **sequence-controlled data processing mode**: The sequence-controlled data processing mode allows the service user to place constraints on each SL-PDU regarding (1) the sequence in which the SL-PDU must be received (with respect to the previous SLPDU) by the Forward Frame service provider and (2) the time window in which the SLPDU must be submitted to production processing. When in the sequence-controlled data processing mode, the Forward Frame service provider validates every SL-PDU to ascertain that the SL-PDU is for the virtual channel that is associated with the Forward Frame service instance and that the SL-PDU is of permitted size and reports back to the service user whether the SL-PDU is valid or invalid (and therefore discarded). For every valid SL-PDU, if the SL-PDU is unable to be processed within the specified time window, the Forward Frame service provider discards the SL-PDU and reports the expiration back to the service user. If the SL-PDU carries the flag to report on completion of data unit processing, the Forward Frame service provider reports back to the service user upon the successful radiation of that SL-PDU;

NOTE – The sequence-controlled data processing mode of the Forward Frame service is not to be confused with the Sequence-Controlled Service of the TC and Unified SDLPs (references [L12] and [L14], respectively). If they are performed, the TC and USLP Sequence-Controlled Services are performed by the MUE of the Earth User Node prior to transfer of the Transfer Frames through the Forward Frame service.

1. **buffered data processing mode**: In order to maximize throughput, the buffered data processing mode does not regulate the flow of SL-PDUs to the underlying production process. Increased throughput is also supported by the capability to place multiple SL-PDUs into a single Forward Buffer in order to maximize the efficiency of the underlying communication service between the Earth User Node and the ESLT. When in the buffered data processing mode, the Forward Frame service provider validates every SL-PDU to ascertain that the SL-PDU is for the virtual channel that is associated with the Forward Frame service instance and that the SL-PDU is of permitted size but reports back to the service user only if that SL-PDU is found to be invalid (and is therefore discarded). If the SL-PDU carries the flag to report on completion of data unit processing, the Forward Frame service provider reports back to the service user upon the successful radiation of that SL-PDU.

The Forward Frame service provides the user with the capability to query or receive periodic reports on the cumulative number of SL-PDUs received by the FF provider instance, the cumulative number of SL-PDUs submitted to production processing, and the number of SLPDUs currently enqueued at the FF provider.

### 2.2 FUNCTIONAL DESCRIPTION

#### 2.2.1 GENERAL

Forward Frame service is provided and produced in one of two configurations, the Transfer Frame configuration or the CADU configuration. Figure 2-1 illustrates the production and provision of the Forward Frame service in the Transfer Frame configuration. Figure 2-2 illustrates the production and provision of the Forward Frame service in the CADU configuration.

Sync and

Channel Coding

Space

Link

Line Legend:

Space link data flow

production parameters, event notifications,

and directives

Aperture

Physical

Channel

T-P

FF-CSTS

Provider

FR instance

(

frames

)

Space Data

Link Protocol

FF-CSTS Production

FF-CSTS Provision

FF-CSTS

Provider

Ports

FF-CSTS

Provider

(

frames

)

T-P

Forward link status

feedback

#### Figure 2-1: Production and Provision of Forward Frame Cross Support Transfer Service—Transfer Frame Configuration

FF-CSTS

Sync and

Channel Coding

Space

Link

Aperture

Physical

Channel

T-P

FF-CSTS

Provider

(

CADUs

)

Provider

Port

FF-CSTS Production FF-CSTS Provision

Line Legend: space link data flow production parameters, event notifications, and directives

#### Figure 2-2: Production and Provision of Forward Frame Cross Support Transfer Service—CADU Configuration

NOTE – The colors of the different components in the figures relate to their specific strata (see 2.2.2.1). Color coding of the strata is used to visually identify the functional categories to which specific transfer services, SDLPs, synchronization and channel coding schemes, etc., belong. This color-coding scheme is used in the example configurations in annex I.

Both figures show two kinds of lines connecting the different components. The double lines connecting through the bodies of the components represent the flow of space link data from the FF-CSTS Provider to the space link. The role of these components in the processing of space link data is the subject of 2.2.2 and 2.2.3.2. The single lines connecting the components from above or below represent the access to the parameters and event notifications of the production functions that the Forward Frame service provides to the service user. The methods of accessing the parameters and events of these production functions are described in 2.2.3.3, 2.2.3.4, 2.2.3.5, and 2.2.3.6.

The following subsections describe the production and provision of the Forward Frame service, respectively.

#### 2.2.2 SERVICE PRODUCTION

##### 2.2.2.1 Service Production in the Transfer Frame Configuration

As depicted in figure 2-1, FF-CSTS production in the Transfer Frame configuration consists of functions in four abstract categories: Aperture, Physical Channel, Synchronization and Channel Coding, and SDLP. These categories are abstract because there are multiple CCSDS Recommended Standards that specify functions that can be combined to support Forward Frame services to user Missions. These abstract categories are called *strata*, and they are analogous to the abstract layers of the International Organization for Standardization (ISO) Open System Interconnection seven-layer reference model (reference [L17]). For example, CCSDS Recommended Standards that define functionality in the Physical Channel stratum include Radio Frequency (RF) and modulation standards for Earth stations (reference [L9] and relay satellites (reference [L10]), and in those for the SDLP stratum include the AOS (reference [L13]), TC (reference [L12]), and USLP (reference [L14]) protocols.

From the perspective of the FF-CSTS Provider, the underlying production functions are performed by *functional resources* (see annex L of reference[1]). Functional resources are the means by which specific instances of functions (e.g., the instance of a Virtual Channel Multiplexing function that forms Master Channel 1 on the S-Band forward link) are configured by Service Management and monitorable (and in some cases controllable) by Cross Support Transfer Services. Functional resources are logical representations of the functionality, but they do not necessarily coincide with physical implementations of that functionality. This logical representation allows different implementations to be mapped onto the same sets of functional resources for purposes of cross-support configuration, monitoring and control.

Different sets of functional resources represent the functions of specific CCSDS space data link protocols, synchronization and channel coding schemes, and RF modulation standards. For example, there is a set of functional resources that represent the TC Space Data Link Protocol Sending End functions (reference [L12]) and a different set of functional resources that represent the Unified Space Data Link Protocol Sending End functions (reference [L14]). Both of these functional resource sets reside in the Space Data Link Protocol stratum. The functional resource specification, which include the identification of the CCSDS protocol, coding, and modulation functions that they represent, are registered in the SANA Functional Resource Registry (reference [4]).

For clarity and simplicity, the descriptions of the production functions that are performed in conjunction with the Forward Frame service are phrase in terms of the functions as they are defined in the source CCSDS Recommended Standards, without reference to the functional resources that represent those functions. Functional resources are mentioned only when they are pertinent to the discussion.

In the Transfer Frame configuration, each FF-CSTS Provider instance transfers the frames of a single VC, and multiple FF-CSTS Provider instances can be used to transfer the frames from various VCs destined for the same physical channel. The multiplexing functions of the SDLP multiplex the Transfer Frames from the Forward Frame service instances that are associated with the same single forward space link physical channel.

When a Forward Frame service is instantiated in a Service Package, it is configured to operate with only one Virtual Channel Multiplexing function which is specific to one SDLP; that is, a Forward Frame service instance can support the frames of only one SDLP during the execution of a Service Package.

NOTE – The multiplexing functions of the CCSDS SDLPs are not limited to multiplexing only VCs carried by Forward Frame service instances. VC Transfer Frames from other sources (e.g., the CCSDS SLE Forward Space Packet service, reference [L19]) can be multiplexed with VC frames carried by one or more Forward Frame service instances into the same single forward space link physical channel. However, such other sources of VC Transfer Frames are outside the scope of this Recommended Standard.

The single multiplexed stream of Transfer Frames is then sent to the Synchronization and Channel Coding functions, which encode the Transfer Frames and append channel synchronization symbols that are appropriate for that particular space link.

For TC and variable-length USLP Transfer Frames, the Synchronization and Channel Coding functions also optionally regulate the transmission of the Transfer Frames based on forward link status information received on the return link from the Space User Node, represented by the dashed arrow in figure 2-1 (see 2.5.2 for more details).

From the Synchronization and Channel Coding functions the physical channel symbol stream goes to the Physical Channel functions that modulate the signal onto a carrier for radiation by the Aperture.

The Forward Frame service instances, the multiplexers, the synchronization and channel coding functions, physical channel functions, and the aperture are configured appropriately by Service Management to provide the required combination of services.

Although designed to work with the SDLP, Synchronization and Channel Coding, Physical Channel, and Aperture functions associated with the TC, AOS, and USLP protocols, the Forward Frame service is capable of working with service production functions in these four strata (e.g., functions associated with the sending side of a specific SDLP) that meet the requirements specified in G2.

##### 2.2.2.2 Service Production in the CADU Configuration

In the CADU configuration, a single FF-CSTS Provider transfers all of the CADUs destined for a given forward space link physical channel. With no need to multiplex frames from different sources, the single FF-CSTS Provider sends its CADUs directly to the Synchronization and Channel Coding functions that have been configured appropriately by Service Management for that space link, as depicted in figure 2-2. The resulting physical channel symbol stream goes to the Physical Channel functions that modulate the signal on a carrier for radiation by the Aperture, just as in the case of the Transfer Frames configuration.

Although designed to work with the Synchronization and Channel Coding, Physical Channel, and Aperture functions associated with the CCSDS CADU processing and transmission, the Forward Frame service is capable of working with service production functions in these three strata that meet the requirements specified in G3.

#### 2.2.3 SERVICE PROVISION

##### 2.2.3.1 General

Provision of the Forward Frame service comprises

1. transfer and validation of SL-PDUs (i.e., Transfer Frames or CADUs);
2. cyclic reporting of parameters of the Forward Frame service provider and the underlying production processing functions;
3. query of the current values of parameters of the Forward Frame service provider and the underlying production processing functions;
4. notification of events related to the Forward Frame service provider and the underlying production processing functions; and
5. ability to invoke real-time reconfiguration of underlying production processing functions.

##### 2.2.3.2 Validation and Queuing of SL-PDUs

###### 2.2.3.2.1 General

The behavior of the Forward Frame service depends on the *data processing mode* of the service instance.

###### 2.2.3.2.2 Validation and Queuing of SL-PDUs in the ‘Sequence-Controlled’ Data Processing Mode

2.2.3.2.2.1 General

When the Forward Frame service instance is configured to operate in the ‘sequencecontrolled’ data processing mode, the prime procedure is the Sequence-Controlled Frame Data Processing procedure, which enforces strict sequencing of the Transfer Frames and provides feedback to the user on the processing status of each SL-PDU.

2.2.3.2.2.2 SL-PDU Validation in the ‘Sequence-Controlled’ Data Processing Mode

Upon receipt of an SL-PDU, the Forward Frame service provider performs the following validation checks:

a) bit-masking the first four octets of each PDU and comparing the result to a bit pattern that represents the data channel identifier for the data channel that is assigned to that Forward Frame service instance, as specified in 4.5.3.1;

NOTES

1. When the Forward Frame service is used to carry Transfer Frames, the bit-mask corresponds to the header fields that contain the GVCID for the specific space data protocol, and the bit pattern corresponds to the specific GVCID assigned to that service instance. When the Forward Frame service is used to carry CADUs, both the bit-mask and the bit pattern are set to ‘all zeroes’ (i.e., ‘don’t care’) since the service instance is the only source of SL-PDUs on the forward space link.
2. Use of the bit-mask-and-compare approach to VC identification provides flexibility to allow the Forward Frame service to support both future CCSDS space data link protocols and non-CCSDS space data link protocols , as long as the identification of the ‘channel’ is contained within the first four octets of the SL-PDU. Both the bit-mask and bit pattern are configured by Service Management for each Forward Frame service instance.
3. Setting the bit mask and bit pattern to ‘all zeroes’ allows the Forward Frame service (in the CADU configuration) to be used to transfer opaque blocks of octets, for example, in support of non-CCSDS-conformant link PDUs that do not necessarily conform to any CCSDS frame and/or coding standards.
4. checking the size of each Transfer Frame against the allowed size, or range of sizes, (as appropriate to the SDLP) configured for that particular service instance;
5. confirming that the data-unit-id value that accompanies the SL-PDU is one greater than that of the previously received SL-PDU;
6. confirming that time of receipt of the SL-PDU is compatible with the earliest-dataprocessing-start-time and latest-data-processing-start-time values that accompany the SL-PDU;
7. confirming that the production-status is either ‘operational’ or ‘interrupted’ at the time of receipt of the SL-PDU; and
8. confirming that the data processing procedure is not in the locked substate.

If the SL-PDU is valid in all of these respects, the Forward Frame service provider accepts the SL-PDU and returns a positive result to the service user. If the SL-PDU fails any of these validity checks, the Forward Frame service provider rejects the SL-PDU and returns a negative result to the service user.

2.2.3.2.2.3 SL-PDU Queuing in the ‘Sequence-Controlled’ Data Processing Mode

Once an SL-PDU is accepted, it enters the Input Queue of the Forward Frame service instance to await release to the underlying production processing. It may await release for reasons such as interruption of the underlying production processing (or when the service is provided in the Transfer Frame configuration, because the VC carried by the Forward Frame service instance has a low priority relative to other VC sources). If an accepted SL-PDU expires while awaiting release to production processing, the frame is discarded, the service instance is locked, and the service user is notified of the expiration of the data unit. The user must either invoke a reset directive or stop and restart the data processing procedure in order to unlock the procedure and resume transfer of data.

###### 2.2.3.2.3 Validation and Queuing of SL-PDUs in the ‘Buffered Data’ Data Processing Mode

2.2.3.2.3.1 General

When the Forward Frame service instance is configured to operate in the ‘buffered data’ data processing mode, the prime procedure is the BFDP procedure. In contrast to the SequenceControlled Frame Data Processing procedure, the BFDP procedure imposes no sequencing or absolute processing time window constraints on the SL-PDUs, and it provides feedback to the user regarding the validity of individual SL-PDUs only when SL-PDUs are found to be invalid (and are therefore discarded).

The buffered data processing mode also allows for the aggregation of multiple SL-PDUs into a single Forward Buffer (from which the data processing mode and the associated data processing procedure get their names). Under the proper circumstances, the buffering of multiple SL-PDUs in a single Forward Buffer can be used to optimize throughput through the underlying data communication service that connects the Forward Frame service provider and the service user, and between the underlying communication service and the Forward Frame service provider entity, but otherwise, whether the SL-PDUs are transferred individually or in buffered groups is not perceptible to the service user.

Together, the absence of sequence control, absence of feedback on valid SL-PDUs, and the use of the Forward Buffer allow for increased throughput and thus the use of higher data rates on the forward space link.

In the ‘buffered data’ data processing mode, the data processing procedure supports two *transfer modes*, ‘complete’ and ‘timely’.

In the ‘complete’ transfer mode, once the data processing procedure has read an SL-PDU from the underlying terrestrial communication service and validated it, the procedure retains the SLPDU for as long as necessary to process and submit it to the underlying service production processing, as described in 2.2.3.2.3.3. If there is no capacity within the procedure instance to receive more SL-PDUs, the procedure suspends reading of new SL-PDUs from the underlying terrestrial communication service until such capacity becomes available.

In the ‘timely’ transfer mode, the data-processing procedure reads and validates every SLPDU from the underlying terrestrial communication service as soon as it is made available by that service. If the accepted SL-PDUs exceed the available capacity of the procedure instance, the procedure discards the oldest unprocessed SL-PDUs, as described in 2.2.3.2.3.3.

2.2.3.2.3.2 SL-PDU Validation in the ‘Buffered Data’ Data Processing Mode

Upon reading of an SL-PDU from the underlying terrestrial communication service, the Forward Frame service provider performs the following validation checks:

a) bit-masking the first four octets of each PDU and comparing the result to a bit pattern that represents the data channel identifier for the data channel that is assigned to that Forward Frame service instance, as specified in 5.5.3.1;

NOTES

1. When the Forward Frame service is used to carry Transfer Frames, the bit-mask corresponds to the header fields that contain the GVCID for the specific space data protocol, and the bit pattern corresponds to the specific GVCID assigned to that service instance. When the Forward Frame service is used to carry CADUs, both the bit-mask and the bit pattern are set to ‘all zeroes’ (i.e., ‘don’t care’) since the service instance is the only source of SL-PDUs on the forward space link.
2. Use of the bit-mask-and-compare approach to VC identification provides flexibility to allow the Forward Frame service to support future CCSDS and nonCCSDS space data link protocols, as long as the identification of the ‘channel’ is contained within the first four octets of the SL-PDU. Both the bit-mask and bit pattern are configured by Service Management for each Forward Frame service instance.
3. Setting the bit mask and bit pattern to ‘all zeroes’ allows the Forward Frame service (in the CADU configuration) to be used to transfer opaque blocks of octets, for example, in support of non-CCSDS-conformant link PDUs that do not necessarily conform to any CCSDS frame and/or coding standards.

b) checking the size of each Transfer Frame against the allowed size, or range of sizes, (as appropriate to the SDLP) configured for that particular service instance.

If the SL-PDU is valid in both these respects, the Forward Frame service provider accepts the SL-PDU. If the SL-PDU fails any of these validity checks, the Forward Frame service provider rejects the SL-PDU and returns a notification to the service user identifying the rejected SL-PDU and the reason for rejection.

2.2.3.2.3.3 SL-PDU Queuing in the ‘Buffered Data’ Data Processing Mode

Once an SL-PDU is accepted, it enters the Input Queue of the Forward Frame service instance to await release to the underlying production processing.

If the service instance is configured to be in the ‘complete’ transfer mode, there are no limitations on the time that the SL-PDU may reside in the Input Queue. If the Input Queue becomes full (e.g., because the underlying service production processing is interrupted), the Forward Frame service provider suspends reading incoming new SL-PDUs from the underlying communication service that connects the Forward Frame service provider to the service user and resumes reading when the Input Queue clears enough space to hold additional SL-PDUs.

If the service instance is configured to be in the ‘timely’ transfer mode, an SL-PDU may be discarded if (a) the SL-PDU has resided in the Input Queue longer than the processinglatency-limit specified for that Forward Frame service instance, or (b) the Input Queue is full and new SL-PDUs arrive, forcing the discarding of an equivalent number of the oldest SL-PDUs.

##### 2.2.3.3 Cyclic Reporting

The service user can use the CSTS Framework Cyclic Report procedure of the Forward Frame service to receive periodic reports containing the current values of configuration parameters, counters, and status indicators associated with the Forward Frame service provision and production.

Because the Forward Frame service is intended to be used with multiple production configurations and is also capable of being used with as-yet-undefined production configurations that perform the functions and meet the requirements specified in annex G, the Forward Frame service does not explicitly ‘know’ about specific configuration parameters, counters, or status indicators of the underlying production processes. Instead, it makes use of the CSTS Cyclic Report procedure’s ability to access those parameters by using the standard names that are assigned to them in the SANA Functional Resources registry (reference [4]).

NOTE – The Functional Resources registry groups the various functions of space communication and navigation production processing and transfer service provision into abstract FR types. For each FR type, configuration parameters, monitorable parameters, notifiable events, and directives (i.e., real-time control commands) are specified. The monitor and control parameters, event notifications, and real-time control capabilities of real systems that implement some part of space communication and navigation production processing and transfer service provision functions are expected to map into the standard parameters, events, and directives of the FRs that correspond to those functions. For readers familiar with the Transmission Control Protocol/Internet Protocol (TCP/IP) Simple Network Management Protocol (SNMP) Management Information Base (MIB) (reference [L18]), the SANA Functional Resources registry is analogous to the SNMP MIB.

##### 2.2.3.4 Information Query

The service user can use the CSTS Framework Information Query procedure of the Forward Frame service to query the current values of configuration parameters, counters, and status indicators associated with the Forward Frame service provision and production. As with the Cyclic Report procedure (2.2.3.3), the Forward Frame service makes use of the Information Query procedure’s ability to access the parameters of the underlying production processing functions by using the standard parameter names that are assigned to them in the SANA Functional Resources registry.

##### 2.2.3.5 Notifications

The service user can use the CSTS Framework Notification procedure of the Forward Frame service to subscribe to event notifications associated with the Forward Frame service provision and production by using the standard event names that are assigned to them in the SANA Functional Resources registry.

##### 2.2.3.6 Real-time Reconfiguration of Production Processing Functions

Some space link service providers allow limited real-time modification of some of the configuration parameters of production processing, for example, bit rate and modulation index. When space link service providers support such real-time control, the optional Master Throw Event procedure of the Forward Frame service (which is derived from the CSTS Framework Throw Event procedure) is available.

The service user can use the Master Throw Event procedure of the Forward Frame service to invoke directives on production processing functions by using the standard directive names that are assigned to them in the SANA Functional Resources registry. The scope of the Master Throw Event procedure is confined to the production processing functions that are directly supporting the instance of the Forward Frame service that executes that procedure.

When the Forward Service is used in the Transfer Frame configuration, there may be multiple Forward Frame service instances, and therefore potentially multiple users that could invoke real-time reconfigurations through the Master Throw Event procedure, but allowing multiple users to change the shared production functions could wreak havoc. Typically, only one service user would have the authority to invoke real-time reconfigurations of the shared production functions. The limitation on such access is enforced through the masterthrow-event-enabled configuration parameter of the Master Throw Event procedure. This parameter, which is configured by Service Management, determines whether the user of the service instance is authorized to use the Master Throw Event procedure. Attempts to invoke directives through a Forward Frame service instance for which Master Throw Event is disabled results in the rejection of those directives.

### 2.3 SERVICE MANAGEMENT

#### 2.3.1 GENERAL

Service Management specifies the set of FF-CSTS Provider instances and supporting production functions to be performed as part of a Service Package, and determines the schedule for the activation and deactivation of those instances and functions.

NOTES

1. A Service Package may include other services in addition to Forward Frame services, and in such cases Service Management specifies and schedules the service instances and production functions related to those services as well. However, discussion of services other than Forward Frame is outside the scope of this Recommended Standard.
2. The means by which Service Management schedules the Service Packages is outside the scope of this Recommendation. The CCSDS Space Communication Cross Support Service Management suite ([L5], [L6], and [L7]) defines a standard set of Service Management information entities used in the scheduling of Service Packages.

#### 2.3.2 INITIAL CONFIGURATION PARAMETER VALUES

Service Management sets the initial values of the configuration parameters of the Forward Frame service instances and the supporting production functions. Some of the configuration parameters of the Forward Frame service may be subsequently modified by the user of the service instance when the service instance is in the inactive state.

The values of configuration parameters of production functions may be dynamically modified by users of Forward Frame service that are permitted to invoke directives during the execution of the Service Package. The specific configuration parameters that are dynamically modifiable for a particular Provider CSSS are defined by that Provider CSSS.

#### 2.3.3 CONFIGURATION CONSISTENCY

Service Management ensures that the configuration of Forward Frame service instances and underlying production are mutually consistent.

Service Management ensures that the ability to invoke directives on the underlying production functions is enabled for only those Forward Frame service instances for which that capability is authorized.

NOTE – Normally, at most one Forward Frame service instance will be authorized to invoke such directives, but the decision depends on the policies of the Missions and the Provider CSSSes. Also, for some Service Packages, the authority to invoke such directives may be restricted to an instance of another type of service, such as SLE Forward Space Packet [L19] or the future Service Control Cross Support Transfer Service (SC-CSTS).

For Service Packages that provide Forward Frame services using the Transfer Frame configuration, Service Management ensures that

1. each Forward Frame Service Provider instance is assigned and configured to accept a unique Virtual Channel (as identified by its GVCID) for the forward space link physical channel to which that provider instance will provide its Transfer Frames;
2. each Forward Frame Service Provider instance is configured to be in the data processing mode (sequence-controlled or buffered data) that is appropriate to the forward space link physical channel to which that provider instance will provide its Transfer Frames;
3. each Forward Frame Service Provider instance is configured to accept Transfer Frames in the size, or range of sizes, that is appropriate to the forward space link physical channel to which that provider instance will provide its Transfer Frames; and d) each Forward Frame Service Provider instance is configured to provide its Transfer Frames to the VC multiplexing function that forms the MC that corresponds to that provider instance’s GVCID.

For Service Packages that provide Forward Frame services using the CADU configuration,

Service Management ensures that

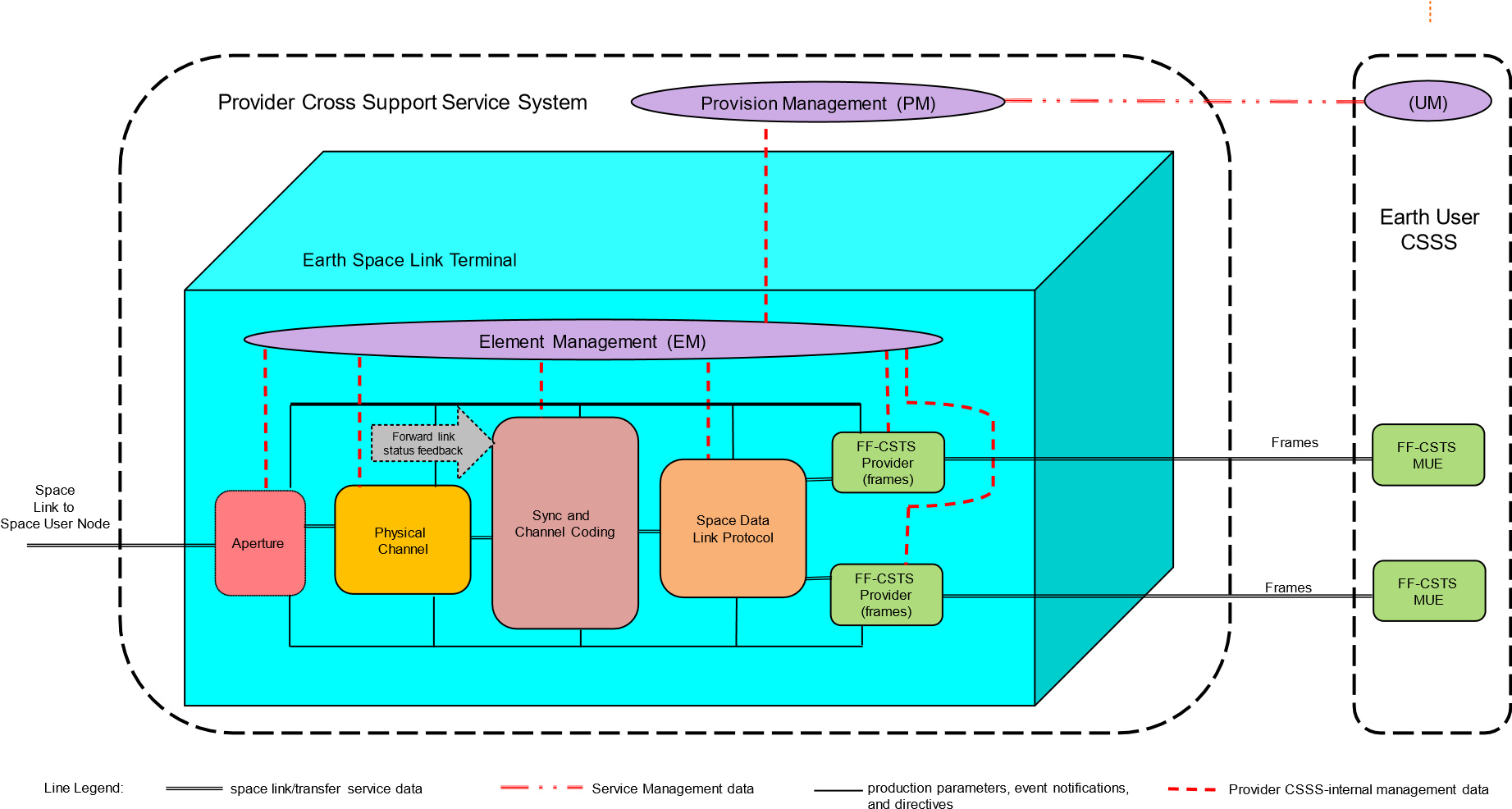
1. the Forward Frame Service Provider instance is configured to ignore the header of the SL-PDUs, as this Forward Frame Service Provider instance is the only one associated with forward space link physical channel and therefore transfers all of the CADUs to that physical channel;
2. the Forward Frame Service Provider instance is configured to be in the data processing mode (sequence-controlled or buffered data) that is appropriate to the forward space link to which that provider instance will provide its SL-PDUs;
3. the Forward Frame Service Provider instance is configured to accept SL-PDUs in the size that is appropriate to the forward space link to which that provider instance will provide its SL-PDUs; and
4. the Forward Frame Service Provider instance is configured to provide its Transfer Frames to the synchronization and channel coding function that forms the physical channel symbol stream for the forward space link.

### 2.4 CROSS SUPPORT VIEW

Figure 2-3 shows an example configuration of an ESLT providing two instances of FF-CSTS to an Earth User CSSS, where the production and provision of the services are in the Transfer Frame configuration. The ESLT is a node of the Provider CSSS.

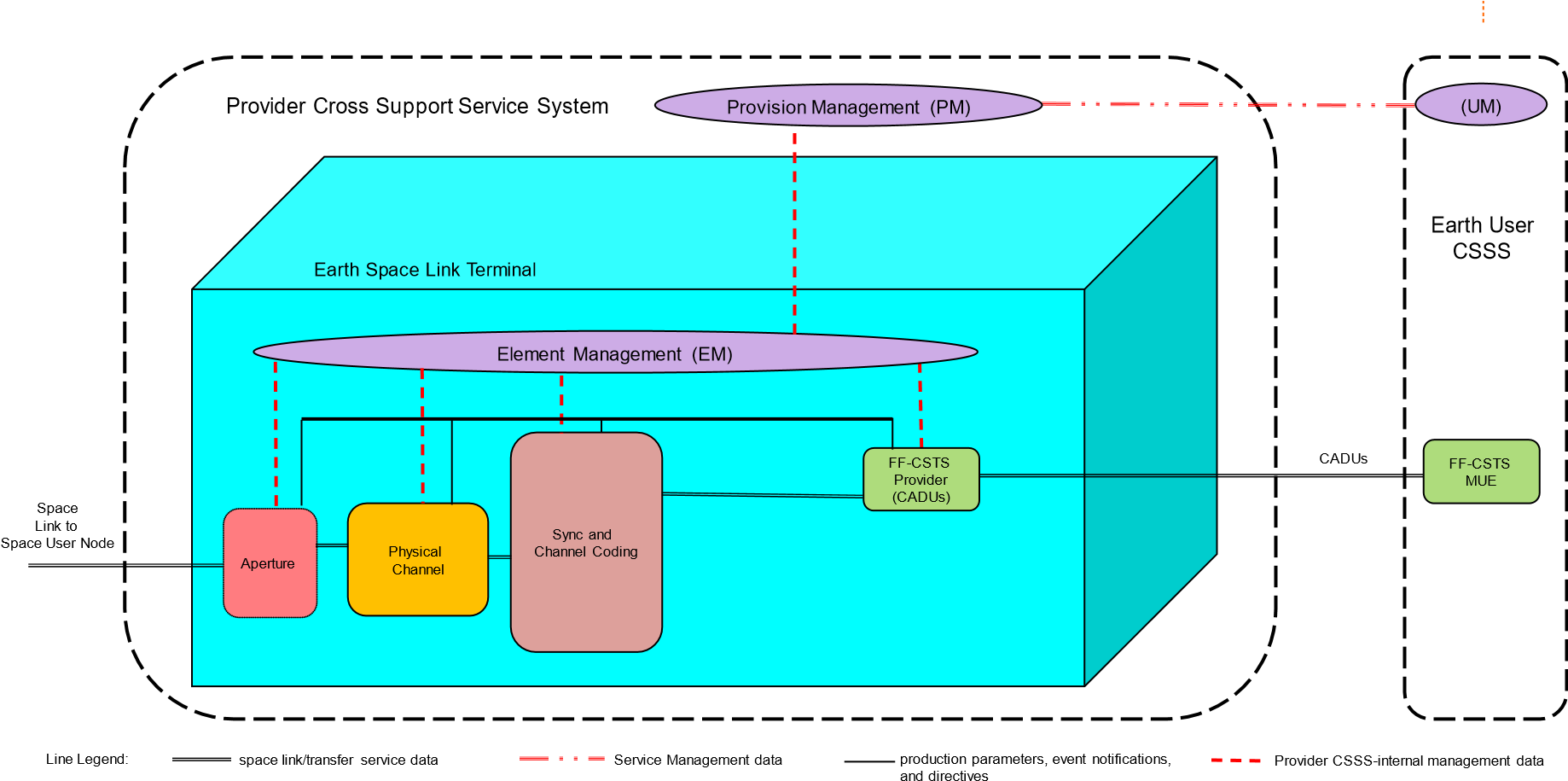
Prior to the execution of the Service Package, the UM function of the Earth User CSSS and the PM function of the Provider CSSS exchange configuration profiles and schedule request information via Service Management, resulting in a scheduled Service Package. Part of the scheduling of the Service Package is the selection of the ESLT that is to execute the Service Package. PM propagates the Service Package information, which includes the information needed to configure the Forward Frame service instances and associated production functions and makes them available to the Mission at the scheduled time, to the EM function of the ESLT assigned to support the Service Package. The EM functions in turn propagate the scheduling and configuration information as needed to the various service production and provision functions that fulfill the Service Package.

During the execution of the Service Package, each FF-CSTS Provider receives Forward Frames from its associated MUE in the Earth User CSSS. Each Forward Frame service provider validates the size and VC of each received frame. Validated frames are input to the SDLP functions, which multiplex the frames from the two Forward Frame service instances. The resultant multiplexed stream of Transfer Frames is then input to the Sync & Channel Coding functions. The resultant encoded physical channel symbol stream is then modulated onto a carrier signal by the Physical Channel function and radiated to the designated Space User Node (e.g., mission spacecraft) by an Aperture for that carrier signal (e.g., an RF antenna).



#### Figure 2-3: Cross Support View of Forward Frame Services—Transfer Frame Configuration

Figure 2-4 shows a similar example configuration of an ESLT providing one instance of FFCSTS to an Earth User CSSS, but in this case the provision and production of the service are in the CADU configuration.



#### Figure 2-4: Cross Support View of Forward Frame Services—CADU Configuration

As described in 2.2.2.2, in the CADU configuration there is only one Forward Frame service instance per forward space link and no SDLP functions are performed.

### 2.5 OPERATIONAL SCENARIOS

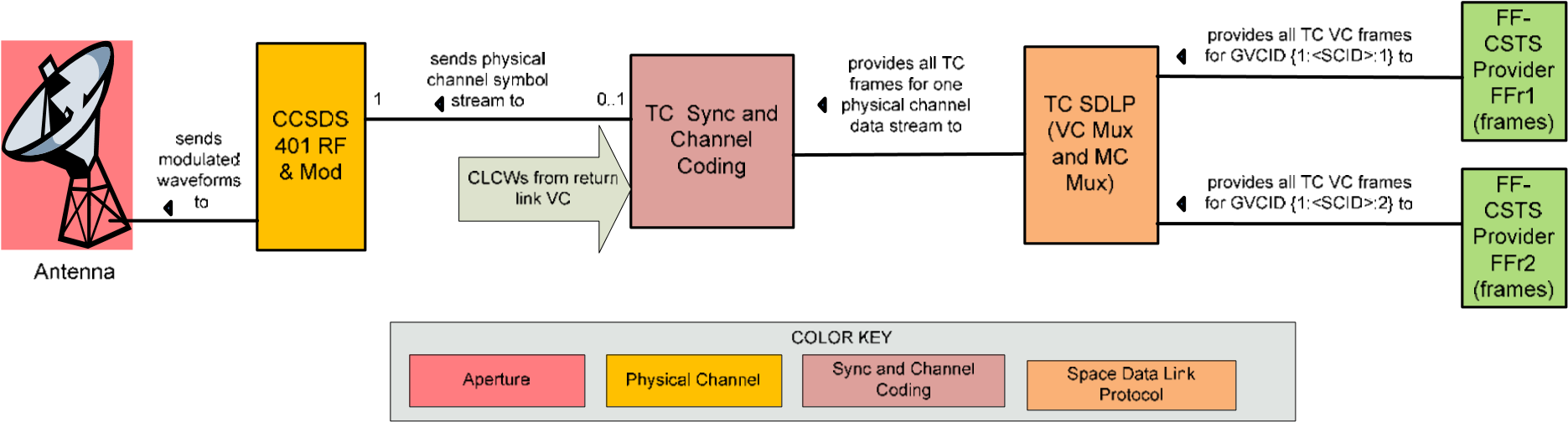
#### 2.5.1 GENERAL

Three operational scenarios are provided in the following subsections. The first scenario illustrates an example of a configuration in which two Forward Frame service instances carry TC frames. The second scenario illustrates an example of a configuration in which two Forward Frame service instances carry AOS frames. The third scenario illustrates an example of a configuration in which one Forward Frame service instance carries CADUs.

NOTE – For the purposes of these example scenarios, the Physical Channel functions are assumed to be performed in accordance with CCSDS 401 (reference [L9]). The Forward Frame service is essentially independent of the functions performed at the Physical Channel stratum. The example scenarios could also use CCSDS 415 (reference [L10]) at this stratum.

#### 2.5.2 TC FRAMES

**2.5.2.1** Figure 2-5 depicts the functions of provision and production components used in the Service Package that supports Transfer Frames over a TC link. In place of the abstract strata designations, the figure identifies the TC SDLP, TC Synchronization and Channel Coding, CCSDS 401 RF and Modulation, and Antenna functions that are performed within those strata.



#### Figure 2-5: Forward Frame Service Scenario—Transfer Frame Configuration— TC Frames

Annex subsection I2 provides a more-detailed description of these functions.

**2.5.2.2** Prior to the provision of service, start and stop times for the space link session and the two Forward Frame service instances are negotiated between PM and UM. Configuration and other information needed to enable the service are also agreed upon, established, and verified. In particular, the following configuration items are established:

1. the Forward Frame service instances FFr1 and FFr2 are configured to be in the ‘sequence-controlled’ data processing mode (3.2), which sets the prime procedure to be the Sequence-Controlled Frame Data Processing procedure;
2. the Forward Frame service instance FFr1 is configured to accept Transfer Frames for TC (that is, version 1) VC 1 for the Mission spacecraft with SCID =20; thus the GVCID of VC 1 in this scenario is {1:20:1}.

Configuring service instance FFr1 to handle this VC consists of

* 1. Setting the gvcid-bit-mask configuration parameter (table 4-3 and 4.7.2) to the TC frame bit mask
     1. 11111111 11111100 00000000 (C3 FF FC 00 Hex);

NOTE – This bit mask corresponds to the presence of the TFVN in bits 0 and 1, the SCID in bits 6–15, and the VCID in bits 16-21 of the first 4 octets of the TC Transfer Frame Primary Header.

* 1. Setting the bit pattern in the authorized-gvcid configuration parameter

(table 4-3 and 4.7.3) to the bit pattern for GVCID {1:20:1}

00000000 00010100 00000100 00000000 (00 14 04 00 Hex); and

* 1. Setting the minimum-frame-length and maximum-frame-length configuration parameters (table 4-3, 4.7.4 and 4.7.5) to the then maximum and minimum values allowed for VC 1;

1. the Forward Frame service instance FFr2 is configured to accept Transfer Frames for VC 2, where the GVCID of VC 2 in this scenario is {1:20:2};
   1. Setting the gvcid-bit-mask configuration parameter (table 4-3 and 4.7.2) to the TC frame bit mask
      1. 11111111 11111100 00000000 (C3 FF FC 00 Hex);
   2. Setting the bit pattern in the authorized-gvcid configuration parameter

(table 4-3 and 4.7.3) to the bit pattern for GVCID {1:20:2}

00000000 00010100 00001000 00000000 (00 14 08 00 Hex); and

* 1. Setting the minimum-frame-length and maximum-frame-length configuration parameters (table 4-3, 4.7.4, and 4.7.5) to the maximum and minimum values allowed for VC 1;

1. the master-throw-event-enabled parameter of service instance FFr1 is set to ‘enabled’, allowing the user of that service instance to invoke directives on the underlying service production;
2. the TC SDLP’s VC Multiplexing function is configured to multiplex VCs 1 and 2 using the absolute priority multiplexing scheme (see G2.2.1.2), with VC 1 having the highest priority and VC 2 having the second-highest priority. For both VC 1 and VC 2, the Number of Repetitions for Sequence-Controlled service (type AD) Frames is configured to be 2, and the Number of Repetitions for COP Control Command (type BC) Frames is configured to be 3 (see reference [L12]);
3. the TC SDLP’s MC Multiplexing function is configured to support the single MC {1:20};

NOTE – Because there is only one MC being used, the MC Multiplexing function does not actually perform any multiplexing; it simply passes along the multiplexed frame stream generated by the VC Multiplexing function.

1. the TC SDLP’s All Frames Generation function is configured to add a Frame Error Control Field (FECF) to the trailer of each frame. The Maximum Number of Transfer Frames Given to the Coding Sublayer as a Single Data Unit is configured to be 5, and the Maximum Length of Data Unit Given to the Coding Sublayer is configured to be 3000 (see reference [L12]).
2. the TC Synchronization and Channel Coding functions are configured to apply frame randomization and Bose–Chaudhuri–Hocquenghem (BCH) encoding of the frame, to perform PLOP-2 (see reference [L11]);
3. the functional resource that represents the TC Synchronization and Channel Coding functions is configured to use the No RF Available and No Bit Lock flags returned from the spacecraft in the Communications Link Control Word (CLCW) fields of the designated return link VC in the determination of the *resource status* of the functional resource;

NOTES

* 1. The use of the CLCW No RF Available and No Bit Lock flags in conjunction with the TC Synchronization and Channel Coding functions is not part of the behavior of those functions as specified in reference [L12]. It is an extended behavior introduced by the SLE Forward CLTU service (see B2.3 of reference [L4]) and the SLE Forward Space Packet service (reference [L19]). The extended behavior is assumed to be performed by the functional resource that represents the TC Synchronization and Channel Coding functions to allow the Forward Frame service to operate in a way that is similar to the FCLTU and FSP services.
  2. The resource-status parameter is common to all production functional resources, and reports whether the functions that are collectively represented by that functional resource have the status of ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’ (see annex B of reference [1]). For CSTSes, the resources statuses of the underlying production functional resources are combined to form the *production status* of the CSTS, which in turn affects the state machine of that CSTS.
  3. Use of the No RF Available flag and/or the No Bit Lock flag of the CLCW to affect the production status of the functional resource is optional and set by Service Management. If neither is configured for use, production status of the functional resource is unaffected by the values of the CLCW flags.
  4. Return link production, which extracts the CLCWs and provides them to the TC Synchronization and Channel Coding function, is not illustrated in figure 2-5.

1. the Forward 401 Space Link Transmission function is configured to apply the selected modulation, frequency, etc., to the input symbol stream;
2. the Antenna function is configured to use the appropriate pointing and frequency data and to operate in the appropriate tracking mode.
   * + 1. At some time before the scheduled start times of the Forward Frame service instances, the details of the Service Package are sent from PM to the EM function of the ESLT. The service instances are created by EM and configured, along with the configuration of the required production functions. Initially, each Forward Frame service provider is in state 1 (‘unbound’). At the scheduled start time of the space link session, the ESLT establishes the forward link to the spacecraft and initiates the production processing for the Forward Frame service instances. Typically (but not necessarily), the start times of the service instances will precede by a small margin the start time of the space link session to allow the user to bind to the service before the start of the space link session.
       2. The following steps illustrate a typical sequence of operations between the user and the provider of either of the Forward Frame service instances configured to transfer TC frames:
   1. The user invokes the BIND operation of the Association Control procedure to establish an association.
   2. The user STARTs the Cyclic Report procedure and specifies which parameters of the service instance and underlying production functions to periodically report and the delivery cycle to be used.
   3. The user STARTs the Notification procedure and subscribes to the ‘tcSyncFwdLinkStatusChange’ event of the resource performing the TC Sync and Channel Coding functions.
   4. The user STARTs the Sequence-Controlled Frame Data Processing procedure, which causes the Sequence-Controlled Frame Data Processing procedure to transition to the ‘active.processing’ substate and the Forward Frame service provider to transition to state ‘bound.active’.
   5. The START invocation identifies the sequence value of the first PROCESS-DATA invocation that will be sent.

NOTES

1. The service user may also use the START invocation to reset the values of certain configuration parameters of the Sequence-Controlled Frame Data Processing procedure, prior to the transfer of Transfer Frames. Activation of the procedure also enables the NOTIFY operation to be invoked when events related specifically to the Sequence-Controlled Frame Data Processing procedure occur.
2. The states of the Sequence-Controlled Frame Data Processing procedure are the same as those of the Framework Sequence-Controlled Data Processing procedure.
3. When production-status changes to ‘operational’ the provider sends a NOTIFY invocation to the user.
4. The user sends a PROCESS-DATA invocation to the provider. The provider verifies that the Transfer Frame contained in the invocation has the correct GVCID, is of the allowed size, and the value of the data-sequence-counter parameter of the PROCESSDATA invocation matches the value that is expected by the service provider:
   1. if the PROCESS-DATA invocation is invalid for any reason, the PROCESSDATA invocation is discarded and a negative result is returned;
   2. if the PROCESS-DATA invocation is valid, the service provider returns a positive result and enqueues the frame.
5. Additional frames are sent by the service user and enqueued if valid.
6. When a frame reaches the front of the queue, it is made available to the VC Multiplexing instance that is associated with that Forward Frame service instance when both (1) the time is equal to or later than that specified by the earliest-data-processstart-time of the frame and (2) production status is operational. If no earliest start time was specified in the PROCESS-DATA invocation that transferred the frame, the frame is made available to production processing as soon as it reaches the front of the queue or whenever the production status becomes operational, whichever is later.
7. If the frame at the front of the queue does not begin to be processed before the latest-data-process-start-time for that frame is reached:
   1. the frame is discarded and a NOTIFY invocation is sent with notification-type ‘expired’;
   2. the Sequence-Controlled Frame Data Processing enters the ‘active.locked’ state, which prevents any further frames in the buffer from being processed and any further PROCESS DATA invocation from being accepted;
   3. upon receipt of the ‘expired’ NOTIFY invocation, the user invokes the EXECUTEDIRECTIVE operation of the Sequence-Controlled Frame Data Processing procedure to ‘reset’ (that is, clear the buffer and begin accepting frames again). Alternatively, the user can invoke the STOP operation, followed by the START operation, which accomplishes the same thing as the ‘reset’ directive.
8. At some time while the service instance is in the ‘bound.ready’ or ‘bound.active’ state, the user invokes the GET operation of the Information Query procedure to retrieve the current value of one or more parameters of that user’s service instance or an underlying production function.
9. At some time while the Notification procedure is active, the CLCWs received from the spacecraft indicate that the No Bit Lock flag has changed from 0 to 1, indicating a loss of bit lock. This causes the resource status of the functional resource that represents the TC Synchronization and Channel Coding functions to change to ‘interrupted’, which in turn causes the production status of the Forward Frame service instance to transition to ‘interrupted’. This causes the Sequence-Controlled Frame Data Processing procedure to enter the ‘active.locked’ substate, discard any frames in the input queue, inhibit receipt of additional frames, and issue a NOTIFY invocation with event ‘production status change’, with event value ‘interrupted’. Concurrently, the Notification procedure issues a NOTIFY invocation for the event ‘telecommandsynchronization-forward-link-status change’, with event value ‘no bit lock’.
10. Once bit lock is re-acquired, the resource status of the functional resource that represents the TC Synchronization and Channel Coding functions changes to ‘operational’, which in turn causes the production status of the Forward Frame service instance to transition to ‘operational’. The Sequence-Controlled Frame Data Processing procedure issues a NOTIFY invocation with event ‘production status change’, with event value ‘operational’. Concurrently, the Notification procedure issues another NOTIFY invocation for the event ‘telecommand-synchronizationforward-link-status change’, this time with event value ‘nominal’.
11. The user of the Forward Frame service invokes the ‘reset’ directive that returns the Forward Frame service instance to the ‘active.processing’ substate and permits the user to resume transferring frames to the service provider.
12. ***For service instance FFr1 only:*** At any time while the service instance is in the ‘bound.ready’ or ‘bound.active’ state, the user may invoke the EXECUTEDIRECTIVE operation of the Master Throw Event procedure to modify the value of one or more configuration parameters of an underlying production function.
13. The user transfers the last frame to the provider.
14. The provider completes processing of the queued frames.
15. The user invokes the STOP operation for the Sequence-Controlled Frame Data Processing procedure and the provider transitions to state 2, ‘bound.ready’.
16. The user invokes the STOP operation for the Cyclic Report procedure to halt cyclic reporting.
17. The user invokes the STOP operation for the Notification procedure to halt event notifications.
18. The user invokes the UNBIND operation to release the association.

**2.5.2.5** During the execution of the Service Package, the production functions perform as follows:

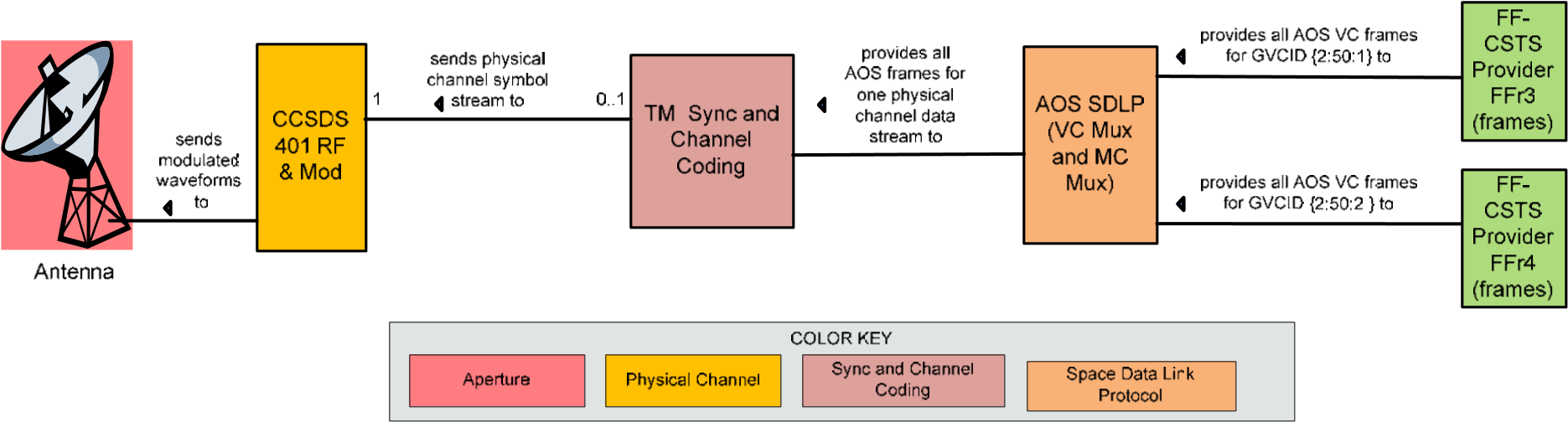
* 1. The VC Multiplexing function polls the queues of both Forward Frame service instances FFr1 and FFr2, with priority to FFr1, pulls the Transfer Frames from the queues accordingly, and multiplexes them into a single MC with MCID {1:20}.
  2. The MC Multiplexing function passes through the multiplexed frames for MC {1:20}.
  3. The All Frames Generation function appends an FECF to each frame and forms groups of frames that conform to the constraints imposed by the configuration parameters Maximum Number of Transfer Frames Given to the Coding Sublayer as a Single Data Unit, and Maximum Length of Data Unit Given to the Coding Sublayer. The resultant groups of frames are submitted to the underlying TC Synchronization and Channel Coding functions in such a way that the CLTU that is formed from each group of frames is transmitted, possibly multiple times, in accordance with configuration parameters that specify the Number of Repetitions for Type-A service (AD) Frames and the Number of Repetitions for COP Control Command (BC) Frames for each virtual channel.
  4. The TC Synchronization and Channel Coding functions randomize and BCH encode the groups of TC frames, and generate CLTUs from them. Each CLTU is subsequently submitted to the PLOP, which appends the appropriate acquisition and idle sequences before and after it.
  5. The resulting symbol stream is modulated onto the RF carrier by CCSDS 401 RF and Modulation functions and radiated to the spacecraft by the Antenna.

#### 2.5.3 AOS FRAMES

**2.5.3.1** Figure 2-6 depicts the provision and production components used in the Service Package that supports Transfer Frames over an AOS link. In place of the abstract strata designations, the figure identifies the AOS SDLP, TM Synchronization and Channel Coding, CCSDS 401 RF and Modulation, and Antenna functions that are performed within those strata.

NOTES

1. As of the publication of this Recommended Standard, the Synchronization and Channel Coding sublayer functions used to support fixed-length frames such as AOS and fixedlength USLP frames on the forward link are planned to be specified in a revised version of reference [L15], which will recommend the use of (a) Reed-Solomon encoding for forward link frames; (b) LDPC encoding for forward link frames; and (c) LDPC coding of a stream of SMTFs (1.6.1.6 d)), a process that consists of slicing a stream of SMTFs, encoding each of the resulting slices into an LDPC codeword, and aggregating a specified number of codewords into an LDPC codeblock, each of which is randomized and has a Code Frame Synchronization Marker attached.
2. Annex subsection I3 provides a more-detailed description of these functions.



#### Figure 2-6: Forward Frame Service Scenario—Transfer Frame Configuration—AOS Frames

**2.5.3.2** Prior to the provision of service, start and stop times for the space link session and the two Forward Frame service instances are negotiated between PM and UM. Configuration and other information needed to enable the service are also agreed upon, established, and verified. In particular, the following configuration items are established:

1. The Forward Frame service instances FFr3 and FFr4 are configured to be in the ‘buffered data’ data processing mode (which sets the prime procedure to be the BFDP procedure, see 3.2) and in the ‘timely’ transfer mode (table 5-3 and 5.7.2).
2. The Forward Frame service instance FFr3 is configured to accept Transfer Frames for AOS (that is, version 2) VC 1 for the Mission spacecraft with the SCID = 50. Thus the GVCID of VC 1 in this scenario is {2:50:1}.

Configuring service instance FFr3 to handle this VC consists of

* 1. Setting the gvcid-bit-mask configuration parameter (table 5-3 and 5.7.3) to the AOS frame bit mask

11111111 11111111 00000000 00000000 (FF FF 00 00 Hex);

NOTE – This bit mask corresponds to the presence of the TFVN in bits 0 and 1, the SCID in bits 2–9, and the VCID in bits 10–15 of the first 4 octets of the AOS Transfer Frame Primary Header.

* 1. Setting the bit pattern in the authorized-gvcid configuration parameter

(table 5-3 and 5.7.4) to the bit pattern for GVCID {2:50:1}

* + 1. 10000001 00000000 00000000 (8C 81 00 00 Hex); and
  1. Setting both the minimum-frame-length and maximum-frame-length configuration parameters (table 5-3, 5.7.5, and 5.7.6) to the fixed-length value allowed for VC 1;

1. The Forward Frame service instance FFr4 is configured to accept Transfer Frames for

VC 2, where the GVCID of VC 2 in this scenario is {2:50:2};

Configuring service instance FFr4 to handle this VC consists of:

* 1. Setting the gvcid-bit-mask configuration parameter (table 5-3 and 5.7.3) to the AOS frame bit mask

11111111 11111111 00000000 00000000 (FF FF 00 00 Hex);

* 1. Setting the bit pattern in the authorized-gvcid configuration parameter

(table 4-3 and 5.7.4) to the bit pattern for GVCID {2:50:2}

* + 1. 10000010 00000000 00000000 (8C 82 00 00 Hex); and
  1. Setting both the minimum-frame-length and maximum-frame-length configuration parameters (table 5-3, 5.7.5, and 5.7.6) to the fixed-length value allowed for VC 2;

1. The master-throw-event-enabled parameter of service instance FFr3 is set to ‘enabled’, allowing the user of that service instance to invoke directives on the underlying service production.
2. The AOS SDLP’s VC Multiplexing function is configured to multiplex VCs 1 and 2 using the absolute priority multiplexing scheme (see G2.2.1.2), with VC 1 having highest priority and VC 2 having the second-highest priority.
3. The AOS SDLP’s MC Multiplexing function is configured to support the single MC {2:50}.
4. The AOS SDLP’s All Frames Generation function is configured to add an FECF to the trailer of each frame.

NOTE – Because there is only one MC being used, the MC Multiplexing function does not actually perform any multiplexing; it simply passes along the multiplexed frame stream generated by the VC Multiplexing function, and generates Only Idle Data Frames in order to preserve data continuity when there are no valid frames available for transmission.

1. The TM Synchronization and Channel Coding functions are configured to perform LDPC encoding, frame randomization, and ASM attachment.
2. The Forward 401 Space Link Transmission function is configured to apply the selected modulation, frequency, etc., to the input symbol stream.
3. The Antenna function is configured to use the appropriate pointing and frequency data, and to operate in the appropriate tracking mode.
   * + 1. At some time before the scheduled start times of the Forward Frame service instances, the details of the Service Package are sent from PM to the EM function of the ESLT. The service instances are created by EM and configured, along with the configuration of the required production functions. Initially, each Forward Frame service provider is in state 1 (‘unbound’). At the scheduled start time of the space link session, the ESLT establishes the forward link to the spacecraft and initiates the production processing for the Forward Frame service instances. Typically (but not necessarily) the start times of the service instances will precede by a small margin the start time of the space link session to allow the users to bind to the services before the start of the space link session.
       2. The following steps illustrate a typical sequence of operations between the user and the provider of either of the Forward Frame service instances configured to transfer AOS frames.
4. The user invokes the BIND operation of the Association Control procedure to establish an association.
5. The user STARTs the Cyclic Report procedure and specifies which parameters of the service instance and underlying production functions to periodically report and the delivery cycle to be used.
6. The user STARTs the Notification procedure and subscribes to one or more events of underlying production functions.
7. The user STARTs the BFDP procedure, and the provider instance transitions to state ‘bound.active’. The service user may use the START invocation to reset the values of certain configuration parameters of the BFDP procedure, prior to the transfer of Transfer Frames. Activation of the procedure also enables the NOTIFY operation to be invoked when events related specifically to the BFDP procedure occur.

NOTE – The states of the BFDP procedure are the same as those of the Framework Buffered Data Processing procedure.

1. When production-status changes to ‘operational’ the provider sends a NOTIFY to the user.
2. The user sends a PROCESS-DATA invocation to the provider. The provider verifies that the Transfer Frame contained in the invocation has the correct GVCID and is of the allowed size:
   1. if the Transfer Frame in the PROCESS-DATA invocation is invalid for any reason, the PROCESS-DATA invocation is discarded and a NOTIFY invocation is sent to the user with the appropriate event and event value;
   2. if the PROCESS-DATA invocation is valid, the service provider enqueues the frame.

NOTE – The BFDP procedure allows PROCESS-DATA invocations (each of which carries one Transfer Frame) to be transferred individually or in multiples in Forward Buffers. For this scenario, it is assumed that the PROCESS-DATA invocations (and therefore the Transfer Frames) are transferred individually.

1. Additional frames are sent by the service user and enqueued if valid.
2. When a frame reaches the front of the queue, it is made available to the VC Multiplexing instance that is associated with that Forward Frame service instance.
3. At some time during the execution of the Service Package, the production processing of the Transfer Frames is interrupted. The BFDP procedure issues a NOTIFY invocation with event ‘production status change’ with event value ‘interrupted’. The interruption causes production processing to stop accepting PROCESS-DATA invocations from the queue and the queue to fill. Once the queue has filled, the service provider instance ceases accepting PROCESS-DATA invocations from the user.

When the interruption is eventually cleared, production processing is once again able to accept frames from the service instance queue. The BFDP procedure issues a NOTIFY invocation with event ‘production status change’ with event value ‘operational’. Once the queue has been cleared enough to hold additional frames, the service instance resumes accepting new PROCESS-DATA invocations from the service user.

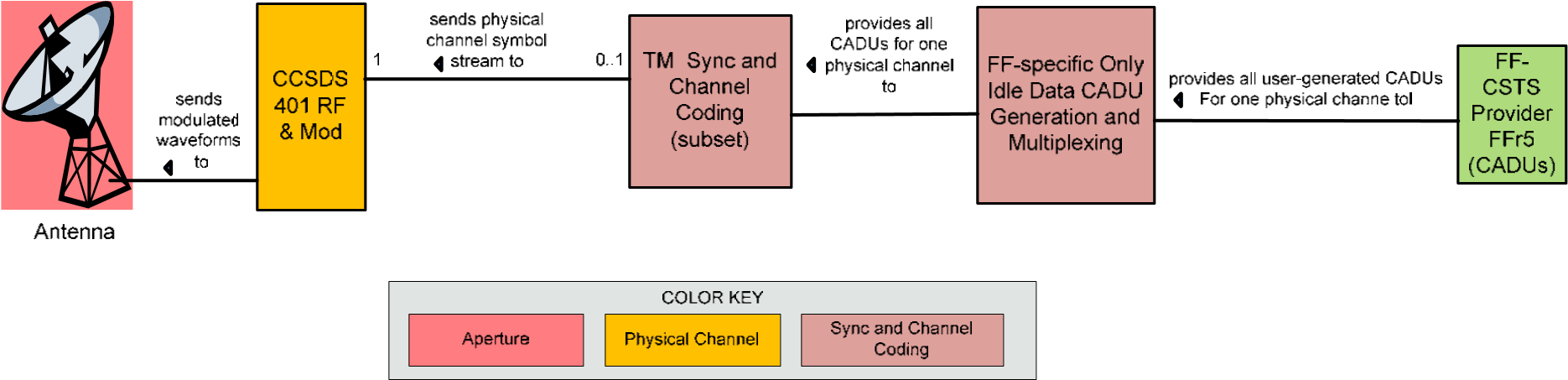
1. At some time while the service instance is in the ‘bound.ready’ or ‘bound.active’ state, the user invokes the GET operation of the Information Query procedure to retrieve the current value of one or more parameters of that user’s service instance or an underlying production function.
2. ***For service instance FFr3 only***: At any time while the service instance is in the ‘bound.ready’ or ‘bound.active’ state, the user may invoke the EXECUTEDIRECTIVE operation of the Master Throw Event procedure to modify the value of one or more configuration parameters of an underlying production function.
3. The user transfers the last frame to the provider.
4. The provider completes processing of the queued frames.
5. The user invokes the STOP operation for the BFDP procedure and the provider transitions to state 2, ‘bound.ready’.
6. The user invokes the STOP operation for the Cyclic Report procedure to halt cyclic reporting.
7. The user invokes the STOP operation for the Notification procedure to halt event notifications.
8. The user invokes the UNBIND operation to release the association.

**2.5.3.5** During the execution of the Service Package, the production functions perform as follows:

1. The VC Multiplexing function polls the queues of both Forward Frame service instances FFr3 and FFr4, with priority to FFr3, pulls the Transfer Frames from the queues accordingly, and multiplexes them into a single MC with MCID {2:50}.
2. The MC Multiplexing function passes through the multiplexed frames for MC {2:50}. If no valid frame is available for transmission at a release time, the Only Idle Data Frame subfunction creates an Only Idle Data frame for release in order to preserve the continuity of the transmitted stream.
3. The All Frames Generation function appends an FECF to each frame.
4. The TM Synchronization and Channel Coding functions LDPC encodes, randomizes, and attaches an ASM to each frame to generate a CADU. The resulting continuous and contiguous stream of CADUs forms the physical channel symbol stream.
5. The physical channel symbol stream is modulated onto the RF carrier by CCSDS 401 RF and Modulation functions and radiated to the spacecraft by the Antenna.

#### 2.5.4 CADUs

**2.5.4.1** Figure 2-7 depicts the provision and production components used in the Service Package that supports CADUs. In place of the abstract strata designations, the figure identifies the TM Synchronization and Channel Coding, CCSDS 401 RF and Modulation, and Antenna functions that are performed within those strata.



#### Figure 2-7: Forward Frame Service Scenario—CADU Configuration

**2.5.4.2** The configuration also includes an Only Idle Data CADU Generation and Multiplexing function that is not defined as part of any other CCSDS space data link protocol or synchronization and channel coding Recommended Standard. This Only Idle Data CADU Generation and Multiplexing function is a unique production processing requirement of the Forward Frame service, as specified in G3.2.1. This function is necessary to generate Only Idle Data CADUs when there are no valid CADUs available for transmission at release time. The generation and multiplexing of Only Idle Data CADUs results in a transmitted waveform that conforms to CCSDS-standard space data link and synchronization and channel coding standard for fixed-length frames. The Only Idle Data CADU Generation and Multiplexing function is logically categorized as belonging to the Synchronization and Channel Coding stratum since its only purpose is to maintain continuous and contiguous synchronization on the physical channel.

NOTE – Annex subsection I6 provides a more-detailed description of these functions.

**2.5.4.3** Prior to the provision of service, start and stop times for the space link session and the Forward Frame service instance are negotiated between PM and UM. Configuration and other information needed to enable the service are also agreed upon, established, and verified. In particular, the following configuration items are established:

1. The single Forward Frame service instance, FFr5, is configured to be in the ‘buffered data’ data processing mode (which sets the prime procedure to be the BFDP procedure, see 3.2) and in the ‘complete’ transfer mode (table 5-3 and 5.7.2).
2. The Forward Frame service instance FFr5 is configured to accept any SL-PDUs of the specified size for CADUs on the target space data link:

Configuring service instance FFr5 to handle CADUs consists of

* 1. Setting the gvcid-bit-mask configuration parameter (table 5-3 and 5.7.3) to the ‘don’t care’ bit mask

00000000 00000000 00000000 00000000 (00 00 00 00 Hex);

* 1. Setting the bit pattern in the authorized-gvcid configuration parameter

(table 5-3 and 5.7.4) to the ‘don’t care’ bit pattern:

00000000 00000000 00000000 00000000 (00 00 00 00 Hex); and

* 1. Setting both the minimum-frame-length and maximum-frame-length configuration parameters (table 5-3, 5.7.5, and 5.7.6) to the fixed-length value of the CADUs.

1. The Forward Frame service instance does not know or care about the data link protocol or coding schemes that have been applied in the CADUs that are being transferred by the service instance. However, the same cannot be said for the Only Idle Data CADU Generation and Multiplexing function, which must inject Only Idle CADUs into the CADU stream that match the space data link protocol and encoding functions applied to the user-generated CADUs so that the resulting waveform is the same as that which would have been created had the CCSDS-standard Only Idle Data Frame generation function been used. In this scenario, the CADUs carry fixed-length USLP frames that are LDPC encoded, randomized, and have had their synch markers attached by the user before they are transferred to the ESLT. The Only Idle Data CADU configuration parameter of the Only Idle Data CADU Generation and Multiplexing function must therefore be configured to contain a USLP Only Idle Data Frame that has been LDPC-encoded, randomized, and synchronization-markered.

NOTE – Because the Only Idle Data CADU has static, pre-determined content, the CADU configuration cannot be used to transfer CADUs that result from the Forward Frame service user using LDPC coding of a stream of Sync-Marked Transfer Frames as defined in reference [L15] (see 1.6.1.6 d)).

1. Because the Forward Frame service user is transferring CADUs, the frame-level coding, optional randomization, and attachment of the synchronization marker functions of TM Synchronization and Channel Coding are not performed by the ESLT as part of the underlying production.
2. The Forward 401 Space Link Transmission function is configured to apply the selected modulation, frequency, etc., to the input symbol stream.
3. The Antenna function is configured to use the appropriate pointing and frequency data, and to operate in the appropriate tracking mode.

**2.5.4.4** At some time before the scheduled start time of the Forward Frame service instance, the details of the Service Package are sent from PM to the EM function of the ESLT. The service instance is created by EM and configured, along with the configuration of the required production functions. Initially, the Forward Frame service provider is in state 1 (‘unbound’). At the scheduled start time of the space link session, the ESLT establishes the forward link to the spacecraft and initiates the production processing for the Forward Frame service instance. Typically (but not necessarily) the start time of the service instance will precede by a small margin the start time of the space link session to allow the user to bind to the service before the start of the space link session.

**2.5.4.5** The following steps illustrate a typical sequence of operations between the user and the provider of the Forward Frame service instance configured to transfer CADUs:

1. The user invokes the BIND operation of the Association Control procedure to establish an association.
2. The user STARTs the Cyclic Report procedure and specifies which parameters of her service instance and underlying production functions to periodically report and the delivery cycle to be used.
3. The user STARTs the Notification procedure and subscribes to one or more events of the underlying production functions.
4. The user STARTs the BFDP procedure, and the provider instance transitions to state ‘bound.active’. The service user may use the START invocation to reset the values of certain configuration parameters of the BFDP procedure, prior to the transfer of CADUs. Activation of the procedure also enables the NOTIFY operation to be invoked when events related specifically to the BFDP procedure occur.

NOTE – The states of the BFDP procedure are the same as those of the Framework Buffered Data Processing procedure.

1. When production-status changes to ‘operational’ the provider sends a NOTIFY invocation to the user.
2. The user sends a PROCESS-DATA invocation to the provider instance. The provider instance verifies that the CADU contained in the invocation is of the allowed size, and
3. if the CADU in the PROCESS-DATA invocation is of the wrong size, the PROCESS-DATA invocation is discarded and a NOTIFY invocation is sent to the user with the appropriate event and event value;
4. if the PROCESS-DATA invocation is valid, the service provider enqueues the CADU.

NOTE – The BFDP procedure allows PROCESS-DATA invocations (each of which carries one CADU) to be transferred individually or in multiples in Forward Buffers. For this scenario, it is assumed that the PROCESS-

DATA invocations (and therefore the CADUs) are transferred individually.

1. Additional CADUs are sent by the service user and enqueued if valid.
2. When a CADU reaches the front of the queue, it is made available to the Only Idle Data CADU Generation and Multiplexing function.
3. At some time during the execution of the Service Package, the production processing of the CADUs is interrupted. The BFDP procedure issues a NOTIFY invocation with event ‘production status change’ with event value ‘interrupted’. The interruption causes production processing to stop accepting PROCESS-DATA invocations from the queue and the queue to fill. Once the queue has filled, the service provider instance ceases accepting PROCESS-DATA invocations from the user.

When the interruption is eventually cleared, production processing is once again able to accept CADUs from the service instance queue. The BFDP procedure issues a NOTIFY invocation with event ‘production status change’ with event value ‘operational’. Once the queue has been cleared enough to hold additional CADUs, the service instance resumes accepting new PROCESS-DATA invocations from the service user.

1. At some time while the service instance is in the ‘bound.ready’ or ‘bound.active’ state, the user invokes the GET operation of the Information Query procedure to retrieve the current value of one or more parameters of that user’s service instance or an underlying production function.
2. At any time while the service instance is in the ‘bound.ready’ or ‘bound.active’ state, the user may invoke the EXECUTE-DIRECTIVE operation of the Master Throw Event procedure to modify the value of one or more configuration parameters of an underlying production function.
3. The user transfers the last CADU to the provider.
4. The provider completes processing of the queued CADUs.
5. The user invokes the STOP operation for the BFDP procedure and the provider transitions to state 2, ‘bound.ready’.
6. The user invokes the STOP operation for the Cyclic Report procedure to halt cyclic reporting.
7. The user invokes the STOP operation for the Notification procedure to halt event notifications.
8. The user invokes the UNBIND operation to release the association.

**2.5.4.6** During the execution of the Service Package, the production functions perform as follows:

* 1. The Forward Frame-specific Only Idle Data CADU Generation and Multiplexing function receives the CADUs directly from the FFr5 service provider. If no user-databearing CADU is available for transmission at a release time, the Only Idle Data CADU Generation and Multiplexing function generates and injects an Only Idle Data CADU (see 2.5.4.3 c)) into the physical channel symbol stream.
  2. The physical channel symbol stream is modulated onto the RF carrier by the RF carrier by CCSDS 401 RF and Modulation functions and radiated to the spacecraft by the Antenna.

## 3 COMPOSITION OF THE FORWARD FRAME CROSS SUPPORT TRANSFER SERVICE

### 3.1 DISCUSSION

The Object Identifiers for the Forward Frame service are specified in annex B.

The Forward Frame service can be implemented as defined herein without need for further extension or refinement.

The Forward Frame service is instantiated in one of two *data processing modes*: sequencecontrolled data processing mode or buffered data processing mode. Each of these modes has its own prime procedure type. Therefore each of these modes is composed of its own set of procedures. Subsection 3.3 specifies the procedures of the sequence-controlled data processing mode of the FF-CSTS, and 3.4 specifies the procedures of the buffered data processing mode of the FF-CSTS.

NOTE – Normally, CSTSes do not have modes that require different prime procedure types. However, because each Forward Frame service instance must support one of several SDLPs, some that require capabilities provided by the SCFDP procedure and some that require the BFDP procedure, the Forward Frame service must allow the SCFDP procedure to be prime in some service instantiations and the BFDP procedure to be prime in other service instantiations. The approach adopted in this specification is to define two modes, each with its own prime procedure. This approach is not documented in the Guidelines (reference [3]) and is used in this service only because the requirements are outside the provisions of the Guidelines. It should be noted that subsection 1.1 of the Guidelines permits such variances as long as they are necessary to the service and are identified in the service specification as being a variance from the Guidelines.

### 3.2 DATA PROCESSING MODE SERVICE MANAGEMENT PARAMETER

**3.2.1** The data processing mode of the Forward Frame service shall set the prime procedure type for the instance of the service, where the ‘sequence-controlled’ value of the mode designates the Sequence-Controlled Frame Data Processing procedure to be the prime procedure type, and the ‘buffered data’ value designates the BFDP procedure to be the prime procedure type.

**3.2.2** The data processing mode of the Forward Frame service shall be set by the service management parameter with the classifier ffDataProcessingMode.

NOTE – The Object Identifiers associated with the service management parameter classifier ffDataProcessingMode, is assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 1} subtree (see annex B).

**3.2.3** The Forward Frame service data processing mode service management parameter shall be of type FfDataProcessingModeType, as specified in the SANA Functional Resource registry.

**3.3 PROCEDURES OF THE SEQUENCE-CONTROLLED DATA PROCESSING**

### MODE OF THE FORWARD FRAME CROSS SUPPORT TRANSFER SERVICE

#### 3.3.1 GENERAL

**3.3.1.1** The sequence-controlled data processing mode of the Forward Frame transfer service shall be composed of the Association Control, Sequence-Controlled Frame Data Processing, Cyclic Report, Notification, Information Query, and Master Throw Event procedures.

**3.3.1.2** There shall be one and only one instance of the Association Control procedure.

**3.3.1.3** The Association Control procedure shall be adopted directly from reference [1].

**3.3.1.4** The version number of the Association Control procedure shall be ‘2’, indicating that the Association Control procedure specified in reference [1] shall be used without substitution of any other version of the Association Control procedure.

**3.3.1.5** The Sequence-Controlled Frame Data Processing procedure shall be the prime procedure for the sequence-controlled data processing mode of the Forward Frame transfer service.

**3.3.1.6** The Sequence-Controlled Frame Data Processing procedure shall be refined and extended from the Sequence-Controlled Data Processing procedure defined in reference [1].

**3.3.1.7** The version number of the Sequence-Controlled Frame Data Processing procedure shall be ‘1’.

**3.3.1.8** There shall be zero secondary instances of the Sequence-Controlled Frame Data Processing procedure.

**3.3.1.9** The Cyclic Report procedure shall be adopted directly from reference [1].

**3.3.1.10** The version number of the Cyclic Report procedure shall be ‘1’, indicating that the Cyclic Report procedure specified in reference [1] shall be used without substitution of any other version of the Cyclic Report procedure.

**3.3.1.11** There shall be zero or more secondary instances of the Cyclic Report procedure.

NOTE – The number of secondary instances of the Cyclic Report procedure is unconstrained by this service specification. However, each real implementation sets the number of instances that will be instantiated.

**3.3.1.12** The Information Query procedure shall be adopted directly from reference [1].

**3.3.1.13** The version number of the Information Query procedure shall be ‘1’, indicating that the Information Query procedure specified in reference [1] shall be used without substitution of any other version of the Information Query procedure.

**3.3.1.14** There shall be zero or one secondary procedure instance of the Information Query procedure.

NOTE – The Information Query procedure is optional; an implementation is not required to include it.

**3.3.1.15** The Notification procedure shall be adopted directly from reference [1].

**3.3.1.16** The version number of the Notification procedure shall be ‘2’, indicating that the Notification procedure specified in reference [1] shall be used without substitution of any other version of the Notification procedure.

**3.3.1.17** There shall be zero or more secondary procedure instances of the Notification procedure.

NOTE – The Notification procedure is optional; an implementation is not required to include it. But if an implementation does include the Notification procedure, it may include one or more instances of it. The number of secondary instances of the Notification procedure is unconstrained by this service specification. However, each real implementation sets the number of instances that will be instantiated.

**3.3.1.18** The Master Throw Event procedure shall be refined and extended from the Throw Event procedure defined in reference [1].

**3.3.1.19** The version number of the Master Throw Event procedure shall be ‘1’.

**3.3.1.20** There shall be zero or one secondary procedure instances of the Master Throw Event procedure.

NOTE – Table 3-1 summarizes the procedures that constitute the sequence-controlled data processing s mode of the Forward Frame transfer service, where (a) the ‘[P]’ in the *Procedure* row designates Sequence-Controlled Frame Data Processing as the prime procedure; (b) *Version* = ‘SFW *v*’ indicates that the version number *v* of the service procedure is the same as that of the CSTS SFW procedure for those procedures that are directly adopted from reference [1] (where *v* = 1 for Cyclic Report and Information Query and *v* = 2 for Association Control and Notification), and *Version* = ‘1’ indicates the version of the refined and extended service procedures (Sequence-Controlled Frame Data Processing and Master Throw Event); (c) *No. of Instances* indicates the minimum and maximum number of allowed instances of each procedure type; (d) *Specification Approach* indicates which procedures are directly adopted or refined and extended; and (e) *Source* indicates the CSTS SFW procedure from which the service procedure is adopted or refined and extended.

#### Table 3-1: Procedures of the Sequence-Controlled Data Processing Mode of the Forward Frame Service

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Procedure | Association Control | SequenceControlled  Frame Data  Processing [P] | Cyclic Report | Information Query | Notification | Master Throw Event |
| Version | SFW 2 | 1 | SFW 1 | SFW 1 | SFW 2 | 1 |
| No. of  Instances | 1..1 | 1..1 | 0..\* | 0..1 | 0..\* | 0..1 |
| Specification  Approach | adopted | refined-andextended | adopted | adopted | adopted | refined and extended |
| Source | Reference [1]:  Association  Control (4.3) | Reference [1]:  Sequence-  Controlled Data  Processing  (4.8) | Reference [1]:  Cyclic Report  (4.10) | Reference [1]:  Information  Query (4.9) | Reference [1]:  Notification  (4.11) | Reference  [1]: Throw  Event (4.12) |

#### 3.3.2 INVOCATION OF THE BUFFERED FRAME DATA PROCESSING PROCEDURE

When the Forward Frame service is configured to be in the Sequence-Controlled data processing mode, the service provider shall

1. respond to a START invocation containing the BFDP procedure-type by rejecting the invocation, returning a START negative result with diagnostic ‘unsupported option’, and continue operating; and
2. respond to any other operation invocation containing the BFDP procedure-type by discarding the invocation and signaling a procedure to association abort with diagnostic ‘protocol error’.

#### 3.3.3 UNIMPLEMENTED OPTIONAL PROCEDURES

##### 3.3.3.1 Discussion

The Cyclic Report, Information Query, Notification, and Master Throw Event procedures of the sequence-controlled data processing mode are optional, meaning implementations may or may not include these procedures. An implementation that does not support an optional procedure rejects incoming invocations to those procedures but does not abort because of them.

##### 3.3.3.2 Unimplemented Cyclic Report procedure

If the implementation of the Forward Frame service does not include the optional Cyclic Report procedure, the service provider shall:

1. respond to a START invocation containing the Cyclic Report procedure-type by rejecting the invocation, returning a START negative result with diagnostic ‘unsupported option’, and continue operating;
2. respond to any other operation invocation containing the Cyclic Report proceduretype by discarding the invocation and signaling a procedure to association abort with diagnostic ‘protocol error’.

##### 3.3.3.3 Unimplemented Information Query procedure

If the implementation of the Forward Frame service does not include the optional Implementation Query procedure, the service provider shall respond to a GET invocation containing the Information Query procedure-type value by rejecting the invocation, returning a GET negative result with diagnostic ‘unsupported option’, and continuing to operate.

##### 3.3.3.4 Unimplemented Notification procedure

If the implementation of the Forward Frame service does not include the optional

Notification procedure, the service provider shall

1. respond to a START invocation containing the Notification procedure-type by rejecting the invocation, returning a START negative result with diagnostic ‘unsupported option’, and continuing to operate; and
2. respond to any other operation invocation containing the Notification procedure-type by discarding the invocation and signaling a procedure to association abort with diagnostic ‘protocol error’.

##### 3.3.3.5 Unimplemented Master Throw Event procedure

If the implementation of the Forward Frame service does not include the optional Master Throw Event procedure, the service provider shall respond to an EXECUTE-DIRECTIVE invocation containing the Master Throw Event procedure-type value by rejecting the invocation, returning an EXECUTE-DIRECTIVE negative acknowledgement with diagnostic ‘unsupported option’, and continuing to operate.

#### 3.4 PROCEDURES OF THE BUFFERED DATA PROCESSING MODE OF THE FORWARD FRAME CROSS SUPPORT TRANSFER SERVICE

##### 3.4.1 GENERAL

**3.4.1.1** The buffered data processing mode of the Forward Frame transfer service shall be composed of the Association Control, BFDP, Cyclic Report, Notification, Information Query, and Master Throw Event procedures.

**3.4.1.2** There shall be one and only one instance of the Association Control procedure.

**3.4.1.3** The Association Control procedure shall be adopted directly from reference [1].

**3.4.1.4** The version number of the Association Control procedure shall be ‘2’, indicating that the Association Control procedure specified in reference [1] shall be used without substitution of any other version of the Association Control procedure.

**3.4.1.5** The BFDP procedure shall be the prime procedure for the buffered data processing mode of the Forward Frame transfer service.

**3.4.1.6** The BFDP procedure shall be refined and extended from the Buffered Data Processing procedure defined in reference [1].

**3.4.1.7** The version number of the BFDP procedure shall be ‘1’.

**3.4.1.8** There shall be zero secondary instances of the BFDP procedure.

**3.4.1.9** The Cyclic Report procedure shall be adopted directly from reference [1].

**3.4.1.10** The version number of the Cyclic Report procedure shall be ‘1’, indicating that the Cyclic Report procedure specified in reference [1] shall be used without substitution of any other version of the Cyclic Report procedure.

**3.4.1.11** There shall be zero or more secondary instances of the Cyclic Report procedure.

NOTE – The number of secondary instances of the Cyclic Report procedure is unconstrained by this service specification. However, each real implementation sets the number of instances that will be instantiated.

**3.4.1.12** The Information Query procedure shall be adopted directly from reference [1].

**3.4.1.13** The version number of the Information Query procedure shall be ‘1’, indicating that the Information Query procedure specified in reference [1] shall be used without substitution of any other version of the Information Query procedure.

**3.4.1.14** There shall be zero or one secondary procedure instance of the Information Query procedure.

NOTE – The Information Query procedure is optional; an implementation is not required to include it.

**3.4.1.15** The Notification procedure shall be adopted directly from reference [1].

**3.4.1.16** The version number of the Notification procedure shall be ‘2’, indicating that the Notification procedure specified in reference [1] shall be used without substitution of any other version of the Notification procedure.

**3.4.1.17** There shall be zero or more secondary procedure instances of the Notification procedure.

NOTE – The Notification procedure is optional; an implementation is not required to include it. But if an implementation does include the Notification procedure, it may include one or more instances of it. The number of secondary instances of the Notification procedure is unconstrained by this service specification. However, each real implementation sets the number of instances that will be instantiated.

**3.4.1.18** The Master Throw Event procedure shall be refined and extended from the Throw Event procedure defined in reference [1].

**3.4.1.19** The version number of the Master Throw Event procedure shall be ‘1’.

**3.4.1.20** There shall be zero or one secondary procedure instances of the Master Throw Event procedure.

NOTE – Table 3-2 summarizes the procedures that constitute the buffered data processing mode of the Forward Frame transfer service, where (a) the ‘[P]’ in the *Procedure* row designates Sequence-Controlled Frame Data Processing as the prime procedure; (b) *Version* = ‘SFW *v*’ indicates that the version number *v* of the service procedure is the same as that of the CSTS SFW procedure for those procedures that are directly adopted from reference [1] (where *v* = 1 for Cyclic Report and Information Query and *v* = 2 for Association Control and Notification), and *Version* = ‘1’ indicates the version of the refined and extended service procedures (BFDP and Master Throw Event); (c) *No. of Instances* indicates the minimum and maximum number of allowed instances of each procedure type; (d) *Specification Approach* indicates which procedures are directly adopted or refined and extended; and (e) *Source* indicates the CSTS SFW procedure from which the service procedure is adopted or refined and extended.

##### Table 3-2: Procedures of the Buffered Data Processing Mode of the Forward Frame Service

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Procedure | Association Control | Buffered Frame  Data  Processing [P] | Cyclic Report | Information Query | Notification | Master Throw Event |
| Version | SFW 2 | 1 | SFW 1 | SFW 1 | SFW 2 | 1 |
| No. of  Instances | 1..1 | 1..1 | 0..\* | 0..1 | 0..\* | 0..1 |
| Specification  Approach | adopted | refined-andextended | adopted | adopted | adopted | refined and extended |
| Source | Reference [1]:  Association  Control (4.3) | Reference [1]:  Buffered Data  Processing  (4.7) | Reference [1]:  Cyclic Report  (4.10) | Reference [1]:  Information  Query (4.9) | Reference [1]:  Notification  (4.11) | Reference [1]:  Throw Event  (4.12) |

##### 3.4.2 INVOCATION OF THE SEQUENCE-CONTROLLED FRAME DATA PROCESSING PROCEDURE

When the Forward Frame service is configured to be in the buffered data processing mode, the service provider shall

1. respond to a START invocation containing the Sequence-Controlled Frame Data Processing procedure-type by rejecting the invocation, returning a START negative result with diagnostic ‘unsupported option’, and continuing to operate; and
2. respond to any other invocation containing the Sequence-Controlled Frame Data Processing procedure-type by discarding the invocation and signaling a procedure to association abort with diagnostic ‘protocol error’.

##### 3.4.3 UNIMPLEMENTED OPTIONAL PROCEDURES

###### 3.4.3.1 Discussion

The Cyclic Report, Information Query, Notification, and Master Throw Event procedures of the sequence-controlled data processing mode are optional, meaning implementations may or may not include these procedures. An implementation that does not support an optional procedure rejects incoming invocations to those procedures but does not abort because of them.

###### 3.4.3.2 Unimplemented Cyclic Report Procedure

Subsection 3.3.3.2 specifies the behavior of an implementation of the Forward Frame service that does not include the optional Cyclic Report procedure.

###### 3.4.3.3 Unimplemented Information Query Procedure

Subsection 3.3.3.3 specifies the behavior of an implementation of the Forward Frame service that does not include the optional Information Query procedure.

###### 3.4.3.4 Unimplemented Notification Procedure

Subsection 3.3.3.4 specifies the behavior of an implementation of the Forward Frame service that does not include the optional Notification procedure.

###### 3.4.3.5 Unimplemented Master Throw Event Procedure

Subsection 3.3.3.5 specifies the behavior of an implementation of the Forward Frame service that does not include the optional Master Throw Event procedure.

#### 3.5 FORWARD FRAME CROSS SUPPORT TRANSFER SERVICE STATE MACHINE

The FF-CSTS state machine shall conform to the state machine for a CSTS with a stateful prime procedure instance, as defined in the CSTS Framework (G3 of reference[1]).

## 4 SEQUENCE-CONTROLLED FRAME DATA PROCESSING PROCEDURE

### 4.1 DISCUSSION

#### 4.1.1 PURPOSE

The SCFDP procedure of the Forward Frame service is used to transfer SDLP Transfer Frames of one VC from the service user to the service provider.

The principal operational philosophy of the SCFDP procedure is that the sequencing and timing of the transmission of each frame is of critical importance, and that if each frame cannot be transmitted in-sequence and within its permitted time window then the transmission of frames must be suspended until proper sequencing and timing can be restored.

NOTE – The SCFDP procedure is derived from the CSTS SFW Sequence-Controlled Data Processing procedure (4.8 of reference [1]). The SCFDP procedure inherits from that Sequence-Controlled Data Processing procedure the following behavior: (a) data units may be accompanied by earliest and latest processing time parameters; (b) the earliest processing time regulates when the data unit can begin to be processed; (c) if the data unit does not begin to be processed by the latest processing time, that data unit and all data units received after it are discarded; and (d) the procedure does not resume accepting data units until the procedure is reset by the service user. This sequence and timing enforcement by the SCFDP should be taken into account when determining the effects on the behavior of protocol functions that produce the frames being transferred by Forward Frame service instances. For example, the Telecommand space data link protocol (reference [L12]) allows frames to be tagged to identify them as frames that bypass the retransmission queue of the COP retransmission protocol. However, the Forward Frame service makes no distinction between the different types of frames. As a result, any expedited TC frames sent via the Forward Frame service in sequence-controlled data processing mode will still be subject to the timing constraints on any frames preceding it, up to and including deletion if one or more of the preceding frames are themselves deleted. Implementations of systems that use the Forward Frame in the sequence-controlled data processing mode should be designed with this behavior in mind.

#### 4.1.2 CONCEPT

The concept of the SCFDP procedure is the same as that of the parent CSTS SFW SequenceControlled Data Processing procedure with additional data validation criteria that result in PROCESS-DATA negative result returns, and addition of a START invocation parameter.

Additional data validation. In addition to the PROCESS-DATA invocation acceptance criteria described in 4.8.2.2 of reference [1], the SCFDP procedure also validates the length and first four octets of the data parameter of the PROCESS-DATA invocation to ensure that the data parameter of the invocation is carrying a valid Transfer Frame for the specific VC and SDLP being supported by the Forward Frame service instance to which the SCFDP procedure instance belongs.

If the length of the frame in the data parameter is outside the range of the minimum and maximum permitted lengths, the PROCESS-DATA invocation is rejected.

For the purposes of the SCFDP procedure, the pertinent Transfer Frame identification information comprises the TFVN (which identifies the SDLP), SCID, and VCID fields. Collectively, these three fields comprise the GVCID. The SCFDP procedure is configured to accept the frames with one GVCID. If the GVCID fields of the data parameter do not match their configured values, the PROCESS-DATA invocation is rejected.

NOTE – Assurance that multiple instances of the SCFDP procedure (i.e., that multiple instances of the Forward Frame service) are not configured to permit transfer of frames for the same VC into the same forward space link is the responsibility of Service Management.

Additional Sequence-Controlled Frame Data Processing START invocation parameter. The SCFDP procedure adds the input-queue-size parameter to the START invocation to allow the user of the service instance that contains the SCFDP procedure to change the value of the input-queue-size configuration parameter of the SCFDP procedure instance as part of activating the procedure instance.

### 4.2 PROCEDURE TYPE IDENTIFIER

The procedure type identifier sequenceControlledFrameDataProcessing, as specified in annex B, shall be used for this procedure.

### 4.3 REFINEMENT

The Sequence-Controlled Frame Data Processing procedure shall refine the PROCESSDATA operation of the CSTS SFW Sequence-Controlled Data Processing procedure by setting the type of the data parameter to be an octet string.

### 4.4 EXTENSION

**4.4.1** The Sequence-Controlled Frame Data Processing procedure shall extend the PROCESS-DATA negative result of the Framework Sequence-Controlled Data Processing procedure by adding the ‘invalid GVCID’, ‘frame too short’, and ‘frame too long’ diagnostics.

**4.4.2** The Sequence-Controlled Frame Data Processing procedure shall extend the START invocation of the Framework Sequence-Controlled Data Processing procedure by adding the input-queue-size parameter.

**4.4.3** The Sequence-Controlled Frame Data Processing procedure shall extend the behavior of the CSTS SFW Sequence-Controlled Data Processing procedure by modifying the behavior of the procedure to

1. permit the modification of the input-queue-size configuration parameter as part of the activation of the procedure instance;
2. reject (in addition to the reasons for rejection inherited from the Sequence-Controlled Data Processing procedure) PROCESS-DATA invocations whose data parameter contents (1) do not match the GVCID configured for that Forward Frame service instance, (2) are shorter than the configured minimum length for that Forward Frame service, or (3) are longer than the configured maximum length for that Forward Frame service; and
3. initialize and update parameters that count the number of data units received, the number of data units submitted to processing, and the current size of the Input Queue.

### 4.5 BEHAVIOR

**4.5.1** The Notification of Production Status Changes, Notification of Production Configuration Changes, Locking, Resetting, Stopping, Terminating, and Aborting behaviors of the Sequence-Controlled Frame Data Processing procedure shall be the same as those of the Sequence-Controlled Data Processing procedure specified in subsections 4.8.3.5, 4.8.3.6, 4.8.3.7, 4.8.3.8, 4.8.3.9, 4.8.3.10, and 4.8.3.11, respectively, of reference [1].

**4.5.2** The Starting behavior of the Sequence-Controlled Frame Data Processing procedure shall be the same as that of the Sequence-Controlled Data Processing procedure, as specified in 4.8.3.1 of reference [1], with the addition of the following requirement:

If the value of the input-queue-size parameter of the START invocation is ‘modify’, the value of the input-queue-size procedure configuration parameter of the Sequence-Controlled Frame Data Processing procedure shall be set to the value of the input-queue-size parameter of the START invocation as part of the activation of the procedure.

**4.5.3** The Transfer and Queuing of Data Units behavior of the Sequence-Controlled Frame Data Processing procedure shall be the same as that of the Sequence-Controlled Data Processing procedure, as specified in 4.8.3.2 of reference [1], with the addition of the following requirements:

**4.5.3.1** If the result of the ANDing of the first 4 octets of the data parameter of the PROCESS-DATA invocation with the gvcid-bit-mask configuration parameter does not match the authorized-gvcid configuration parameter, the PROCESS-DATA invocation shall be invalid, and the service provider shall issue a PROCESS-DATA negative return with the diagnostic ‘invalid GVCID’.

1. For Telecommand frames, the value of the gvcid-bit-mask configuration parameter shall be

11000011 11111111 11111100 00000000 (C3 FF FC 00 Hex).

1. For Telecommand frames, the bits of the authorized-gvcid configuration parameter that correspond to the TFVN, SCID, and VCID fields of the Telecommand Transfer Frame Primary Header shall be set to the GCVID of the permitted virtual channel.
2. For AOS frames, the value of the gvcid-bit-mask parameter shall be 11111111 11111111 00000000 00000000 (FF FF 00 00 Hex).
3. For AOS frames, the bits of the authorized-gvcid configuration parameter that correspond to the TFVN, SCID, and VCID fields of the AOS Transfer Frame Primary Header shall be set to the GCVID of the permitted virtual channel.
4. For USLP frames, the value of the gvcid-bit-mask parameter shall be 11111111 11111111 11111111 11100000 (FF FF FF E0 Hex).
5. For USLP frames, the bits of the authorized-gvcid configuration parameter that correspond to the TFVN, SCID, Source or Destination ID, and VCID fields of the USLP Transfer Frame Primary Header shall be set to the GCVID of the permitted virtual channel.

NOTE – Frames transferred by the Forward Frame service contain the SCID of the destination space node. Therefore the Source or Destination ID bit of the authorized-gvcid configuration parameter that corresponds to the Source or Destination ID bit of the USLP Transfer Frame Primary Header (the 21st bit) shall be set to 1 when the Forward Frame service is provided by an ESLT.

**4.5.3.2** If the length of the data parameter of the PROCESS-DATA invocation is less than that specified by the minimum-frame-length configuration parameter, the PROCESSDATA invocation shall be invalid, and the service provider shall issue a PROCESS-DATA negative return with the diagnostic ‘frame too short’.

**4.5.3.3** If the length of the data parameter of the PROCESS-DATA invocation is greater than that specified by the maximum-frame-length configuration parameter, the PROCESS-DATA invocation shall be invalid, and the service provider shall issue a PROCESS-DATA negative return with the diagnostic ‘frame too long’.

**4.5.3.4** Upon successful validation of the PROCESS-DATA invocation, the service

provider shall

1. increment the value of the number-of-data-units-received parameter by 1; and
2. increment the input-queue-size parameter by 1.

**4.5.3.5** For each PROCESS-DATA invocation that is discarded from the Input Queue without being transferred to production processing, the service provider shall decrement the input-queue-size parameter by 1.

**4.5.4** The Processing of Data Units behavior of the Sequence-Controlled Frame Data Processing procedure shall be the same as that of the Sequence-Controlled Data Processing procedure, as specified in 4.8.3.3 of reference [1], with the addition of the following requirements: Upon transfer of the data unit from the Input Queue to service production, the service provider shall

1. increment the value of the number-of-data-units-submitted-toprocessing parameter by 1; and
2. decrement the input-queue-size parameter by 1.

**4.5.5** The Positive Feedback requirements of the Sequence-Controlled Data Processing procedure, as specified in 4.8.3.4 of reference [1], shall be replaced with the following requirement: Upon the receipt of a ‘data unit processing completed’ event from service production, the service provider shall

1. increment the value of the number-of-data-units-processed parameter by 1; and
2. if the value of the process-completion-report parameter (see 4.6.4.1.4 of reference [1]) in the associated PROCESS-DATA invocation was ‘produce report’, send a NOTIFY invocation with Event Identifier ‘data unit processing completed’.

NOTE – The ‘data unit processing completed’ event is emitted by service production at the detected or estimated time of radiation of the data unit (see G2.3.4).

#### 4.5.6 INITIALIZATION OF DATA UNIT COUNTERS

At the start of the service instance provision period, the service provider shall initialize the values of the number-of-data-units-received, number-of-data-unitssubmitted-to-processing, and number-of-data-units-processed parameters to ‘zero’.

### 4.6 REQUIRED OPERATIONS

**4.6.1** The Sequence-Controlled Frame Data Processing procedure shall use the STOP, NOTIFY, and EXECUTE-DIRECTIVE operations of the Sequence-Controlled Data

Processing procedure, as specified in reference [1], without extension or refinement.

**4.6.2** The Sequence-Controlled Frame Data Processing procedure shall extend the START operation of the Sequence-Controlled Data Processing procedure, as specified in 4.6.4.

**4.6.3** The Sequence-Controlled Frame Data Processing procedure shall extend and refine the PROCESS-DATA operation of the Sequence-Controlled Data Processing procedure, as specified in 4.6.4.

NOTE – Table 4-1 summarizes the operations of the Sequence-Controlled Frame Data Processing procedure of the Forward Frame service.

#### Table 4-1: Sequence-Controlled Frame Data Processing Procedure Required Operations

|  |  |  |  |
| --- | --- | --- | --- |
| **Operations** | **Extended** | **Refined** | **Procedure**  **Blocking/Non-Blocking** |
| START | Y | N | Blocking |
| STOP | N | N | Blocking |
| PROCESS-DATA | Y | Y | Non-Blocking |
| NOTIFY | N | N | Non-Blocking |
| EXECUTE-  DIRECTIVE | N | N | Blocking |

#### 4.6.4 START (CONFIRMED)

##### 4.6.4.1 Invocation, Return, and Parameters

###### 4.6.4.1.1 General

In addition to the parameters of the START operation of the Sequence-Controlled Data Processing procedure, defined in 4.8 of reference [1], the extension parameter defined in table 4-2 shall be present in the START invocation of the Sequence-Controlled Frame Data Processing procedure.

#### Table 4-2: START Extension Parameter

|  |  |  |
| --- | --- | --- |
| **Extension Parameter** | **Invocation** | **Return** |
| input-queue-size | M |  |

##### 4.6.4.1.2 Extension Parameter Syntax

The type SequContrFrameDataProcStartInvocExt, as defined in annex C, shall define the syntax of the extension parameter of the START invocation.

##### 4.6.4.1.3 input-queue-size

The input-queue-size parameter shall have one of two values:

1. ‘unchanged’: The value of the input-queue-size configuration parameter of the Sequence-Controlled Frame Data Processing procedure instance is to be unchanged from its current value.
2. ‘modify’: The value of the input-queue-size configuration parameter of the Sequence-Controlled Frame Data Processing procedure instance is to be modified. The ‘modify’ value contains the value (in number of PROCESS-DATA invocations that can be stored in the queue) to which the input-queue-size configuration parameter is to be set.

**4.6.4.1.4** The input-queue-size configuration parameter shall be present in every START invocation.

#### 4.6.5 PROCESS-DATA (CONFIRMED)

##### 4.6.5.1 Invocation, Return, and Parameters

###### 4.6.5.1.1 data Parameter Refinement

The data parameter of the PROCESS-DATA invocation shall be formatted as an octet string.

NOTES

1. The data parameter of the Framework core PROCESS-DATA invocation has an abstract type that must be resolved by the procedure that uses the PROCESS-DATA operation. The Framework Sequence-Controlled Data Processing procedure defers the resolution of the data parameter to the service that uses a procedure that is derived from the Framework Sequence-Controlled Data Processing procedure

(reference [1], 4.8.4.2.1, Note).

1. The data parameter is expected to contain a Transfer Frame for a support CCSDS space data link protocol. However, the Forward Frame will actually validate any octet string with any format as long as (a) the first four octets of the octet string, when ANDed with the gvcid-bit-mask configuration parameter, matches the authorized-gvcid configuration parameter; and (b) the length of the octet string is within the configured minimum and maximum lengths.

###### 4.6.5.1.2 Return diagnostic

4.6.5.1.2.1 General

The diagnostic parameter shall have one of the diagnostic values specified in

4.8.4.2.7.1 of reference [1] or one of the following values:

1. ‘invalid GVCID’—the header fields corresponding to those of the GVCID do not match the authorized GVCID for this procedure instance (see 4.5.3.1);
2. ‘frame too short’—the length of the data parameter of the PROCESS-DATA invocation is less than the minimum frame length specified for this procedure instance (see 4.5.3.2);
3. ‘frame too long’—the length of the data parameter of the PROCESS-DATA invocation is greater than the maximum frame length specified for this procedure instance (see 4.5.3.3).

4.6.5.1.2.2 Return diagnostic extension syntax

The type SequContrFrameDataProcProcDataDiagnosticExt, as defined in annex C, shall define the syntax of the extension diagnostics for the PROCESS-DATA negative return.

### 4.7 CONFIGURATION PARAMETERS

**4.7.1** The Sequence-Controlled Frame Data Processing procedure configuration parameters that need to be configured in the context of the procedure shall be as defined in table 4-3.

NOTES

1. For each configuration parameter, the table provides the engineering unit (if applicable), provides a cross reference to the use of the parameter in the specification of the procedure, identifies whether the parameter may be read and/or dynamically modified, and also identifies the Parameter Identifier and type to be used in reporting the value of the parameter.
2. Section 1.6.1.7.37 of the reference [1] states, ‘Unless explicitly specified otherwise in the Framework definition of a procedure type, the method(s) by which the values of dynamically modifiable procedure configuration parameters are modified is (are) delegated to the service using the procedure or to a derived procedure’. For the Sequence-Controlled Frame Data Processing procedure, the only dynamically modifiable configuration parameter is the input-queue-size parameter. For the Forward Frame service, the only method by which the value of this parameter can be modified is through the START invocation as specified in 4.6.4.1.3.

#### Table 4-3: Sequence-Controlled Frame Data Processing Procedure Configuration Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameters** | **Cross-**  **Reference** | **Readable** | **Dynamically modifiable** | **Configuration Parameter Identifier and Type** |
| input-queue-  size (in number of  PROCESS-DATA  invocations the queue will store) | CSTS SFW  (reference  [1]), 4.6.3.2.3 | Yes | Yes | pDPinputQueueSize PDPinputQueueSizeType  (CSTS SFW F3.16) |
| gvcid-bit-mask  (4-octet) | 4.5.3.1 | Yes | No | pSCFDPgvcidBitMask PSCFDPgvcidBitMaskType  (annex F) |
| authorized-  gvcid (4-octet) | 4.5.3.1 | Yes | No | pSCFDPauthorizedGvcid PSCFDPauthorizedGvcidType  (annex F) |
| minimum-frame-  length (in octets) | 4.5.3.2 | Yes | No | pSCFDPminFrameLen PSCFDPminFrameLenType  (annex F) |
| maximum-frame-  length (in octets) | 4.5.3.3 | Yes | No | pSCFDPmaxFrameLen PSCFDPmaxFrameLenType  (annex F) |

**4.7.2** The gvcid-bit-mask shall be set to the bit mask pattern for the applicable space data link protocol data unit type (TC, AOS, USLP, CADU, or other) that is specified in the service management parameter with the classifier ffAuthorizedGvcidAndBitMask.

**4.7.3** The authorized-gvcid shall be set to the value that is specified in the service management parameter with the classifier ffAuthorizedGvcidAndBitMask.

**4.7.4** The minimum-frame-length shall be set to the value of the service management parameter with the classifier ffMinFrameLength.

**4.7.5** The maximum-frame-length shall be set to the value of the service management parameter with the classifier ffMaxFrameLength.

NOTES

1. The classifier for the input-queue-size configuration parameter is specified in 7.3.
2. If the procedure is used to transfer fixed-length frames, the values of the ffMinFrameLength and ffMaxFrameLength parameters are the same.
3. The Object Identifiers associated with the service management parameter classifiers ffAuthorizedGvcidAndBitMask, ffMinFrameLength, and ffMaxFrameLength are assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 1} subtree (see annex B).

### 4.8 READ-ONLY PARAMETERS

**4.8.1** The Sequence-Controlled Frame Data Processing procedure read-only parameters shall be as defined in table 4-4.

NOTE – For each read-only parameter, the table provides the engineering unit (if applicable), provides a cross reference to the use of the parameter in the specification of the procedure, the classifier of the parameter, the type to be used in reporting the value of the parameter, and the annex in which that type is specified.

#### Table 4-4: Sequence-Controlled Frame Data Processing Procedure Read-Only Parameters

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Cross-Reference** | **Read-Only Parameter Classifier and Type** |
| number-of-dataunits-received (in number of PROCESS-  DATA invocations) | 4.5.3.4 a), 4.5.6 | pSCFDPnumberDataUnitsRcvd PSCFDPnumberDataUnitsRcvdType  (annex F) |
| number-of-dataunits-submittedto-processing (in number of PROCESS-  DATA invocations) | 4.5.4 a), 4.5.6 | pSCFDPnumberDataUnitsToProcessing PSCFDPnumberDataUnitsToProcessingType  (annex F) |
| number-of-dataunits-processed  (in number of  PROCESS-DATA  invocations) | 4.5.5 a), 4.5.6 | pSCFDPnumberDataUnitsProcessed PSCFDPnumberDataUnitsProcessedType  (annex F) |

**4.8.2** When the Sequence Controlled Frame Data Processing Procedure is the prime procedure type, the functional resource parameter (see 1.6.1.12.3) with the classifier ffNumberDataUnitsRcvd shall be set to the value of the SCFDP number-of-dataunits-received parameter.

**4.8.3** When the Sequence Controlled Frame Data Processing Procedure is the prime procedure type, the functional resource parameter with the classifier ffNumberDataUnitsToProcessing shall be set to the value of the SCFDP numberof-data-units-submitted-to-processing parameter.

**4.8.4** When the Sequence Controlled Frame Data Processing Procedure is the prime procedure type, the functional resource parameter with the classifier ffNumberDataUnitsProcessed shall be set to the value of the SCFDP number-ofdata-units-processed parameter.

### 4.9 PROCEDURE STATE TABLE

NOTE – In the following tables, the specifications that are inherited from the parent Sequence-Controlled Data Processing procedure are *italicized*, and additions to the tables that are specific to the Sequence-Controlled Frame Data Processing procedure are in plain text.

#### Table 4-5: Sequence-Controlled Frame Data Processing Procedure State Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Incoming Event** | **State 1**  **(‘inactive’)** | **State 2.1**  **(‘active.processing’)** | **State 2.2**  **(‘active.locked’)** |
| 1 | *(StartInvocation)* | *IF “positive result”*  *THEN*  ‘change the input queue size’  *(+StartReturn)*  *→ 2.1*  *ELSE*  *(-StartReturn)*  *ENDIF* | *{procedure to association abort*  *‘protocol error’} → 1* | *{procedure to association abort ‘protocol error’} → 1* |
| 2 | *(StopInvocation)* | *{procedure to association abort*  *‘protocol error’}* | *IF “positive result”*  *THEN*  *{initiate stop}*  *→ 1*  *ELSE*  *(-StopReturn)*  *ENDIF* | *IF “positive result”*  *THEN*  *{initiate stop}*  *→ 1*  *ELSE*  *(-StopReturn)*  *ENDIF* |
| 3 | *(ProcessDataInvocation)* | *{procedure to association abort*  *‘protocol error’}* | *IF “positive result”*  *THEN*  *‘queue data unit’*  ‘increment data units received’  ‘increment current input queue’  *(+ProcessDataReturn)*  *ELSE*  *(-ProcessDataReturn)*  *ENDIF* | *(-ProcessDataReturn)* |
| 4 | *‘data unit ready’* | *Not applicable* | *IF ‘production status =*  *‘interrupted’’*  *THEN*  *→ 2.2*  *ELSE*  *‘process data unit’*  ‘increment data units submitted to processing’  ‘decrement current input queue’  *ENDIF* | *Not applicable* |
| 5 | *‘data unit processing completed’* | *[ignore]* | *IF “report”*  *THEN*  *‘notify ‘data processing*  *completed’ / ‘empty’’*  *ENDIF* | *Not applicable* |
| 6 | *‘expired’* | *Not applicable* | *‘discard data unit’*  ‘decrement current input queue’  ‘*notify ‘expired’ / ‘empty’’*  *→ 2.2* | *Not applicable* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Incoming Event** | **State 1**  **(‘inactive’)** | **State 2.1**  **(‘active.processing’)** | **State 2.2**  **(‘active.locked’)** |
| 7 | *‘production status change to ‘interrupted’’* | *[ignore]* | *‘notify ‘production status change’ /*  *‘interrupted’’*  *IF ‘processing data unit”*  *THEN*  *‘discard data units in processing’*  *→ 2.2*  *ENDIF* | *[ignore]* |
| 8 | *‘production status change to ‘halted’’* | *[ignore]* | *‘discard data units in processing’*  *‘notify ‘production status change’ /*  *‘halted’’*  *→ 2.2* | *‘notify ‘production status change’ /*  *‘halted’’* |
| 9 | *‘production status change to ‘operational’’* | *[ignore]* | *Not applicable* | *‘notify ‘production status change’ / ‘operational’’* |
| 10 | *‘production status change to ‘configured’’* | *[ignore]* | *Not applicable* | *‘notify ‘production status change’ / ‘configured’’* |
| 11 | *‘production configuration change’* | [ignore] | ‘notify ‘production configuration change’’ | ‘notify ‘production configuration change’/‘empty’’ |
| 12 | *‘data processing configuration change’* | *[ignore]* | *‘notify ‘data processing*  *configuration change’ / ‘procedure configuration parameter values’’ (see 4.8.4.3.4.1 of reference [1])* | *‘notify ‘data processing*  *configuration change’ / ‘procedure configuration parameter values’’ (see 4.8.4.3.4.1 of reference [1])* |
| 13 | *‘invalid PDU ‘xxx’’* | *{procedure to association abort ‘protocol error’}* | *{procedure to association abort*  *‘xxx’} → 1* | *{procedure to association abort*  *‘xxx’} → 1* |
| 14 | *‘terminate procedure’* | *‘terminate itself’* | *‘terminate itself’* | *‘terminate itself’* |
| 15 | *(ExecuteDirectiveInvoca*  *tion)* | *{procedure to association abort ‘protocol error’}* | *IF directive-identifier = ‘reset’*  *THEN*  *(+ExecuteDirective-*  *Acknowledge)*  *{reset}*  *(+ExecuteDirectiveReturn)*  *ELSE*  *{- ExecuteDirective-*  *Acknowledge with diagnostic ‘unknown directive’}* | *IF directive-identifier = ‘reset’*  *THEN*  *(+ExecuteDirective-*  *Acknowledge)*  *{reset}*  *(+ExecuteDirectiveReturn)*  *→ 2.1*  *ELSE*  *{- ExecuteDirective-*  *Acknowledge with diagnostic ‘unknown directive’}* |

#### Table 4-6: Procedure State Table Incoming Event Description References

|  |  |
| --- | --- |
| **Event** | **Reference** |
| *‘expired’* | *(4.8.3.3.2.5 and 4.8.4.3.2 of reference [1]). A data unit is available at the head of the Input Queue, but its latest-data-processstart-time has expired.* |
| *‘data unit processing completed’* | 4.5.5 |
| *‘data unit ready’* | *4.8.3.3.2 of reference [1]*  *A data unit is available at the head of the Input Queue, and production-status is ‘operational’; neither earliest-dataprocess-start-time nor latest-data-process-starttime is specified, or the current time is between earliestdata-process-start-time and latest-data-processstart-time, and production-status is ‘operational’.* |
| *‘terminate procedure’* | *4.2.3 of reference [1]* |
| *‘production status change to ‘xxx’* | *B2.5 of reference [1]* |
| *‘production configuration change’* | *3.11.2.2.3.2 b) of reference [1]* |
| *‘data processing configuration change’* | *4.6.4.2.3 c) of reference [1]* |
| *‘invalid PDU xxx’’* | *3.2.3.6, 4.2.2.4 of reference [1], where ‘xxx’ is one of the diagnostic values specified in 4.2.2.5 of reference [1]* |

#### Table 4-7: Procedure State Table Predicate Descriptions

|  |  |
| --- | --- |
| **Predicate** | **Evaluates to TRUE if** |
| *“positive result”* | *No reason for sending a negative return has been detected; that is, for the START invocation, none of the conditions in (3.7.2.3.1 or 4.6.3.1 a) of reference [1]) apply; for the STOP invocation none of the conditions in 3.3.2.7.1 of reference [1] applies;* and for the PROCESS-DATA invocation none of the conditions in 4.6.5.1.2.1 apply. |
| *“report”* | *The process-completion-report parameter value in the associated (ProcessDataInvocation) PDU is ‘produce report’ (see 4.6.4.1.4 of reference [1]).* |
| *“processing data unit”* | *A data unit has been read from the top of the Input Queue and processing of this data unit has started but not completed* |
| *“production status =*  *‘interrupted’”* | *The current value of production-status is ‘interrupted’.* |

#### Table 4-8: Procedure State Table Simple Action References

|  |  |
| --- | --- |
| **Name** | **References** |
| *‘queue data unit’* | 4.5.3 |
| *‘process data unit’* | 4.5.4 |
| *‘complete data processing’* | *4.6.3.6.2 b) of reference [1]* |
| *‘clear the Input Queue’* | *(4.6.3.6.2 a) and 4.8.3.8 b) of reference [1]).* |
| *‘notify ‘xxx’ / ‘yyy’’* | *(4.6.3.5.1.1, 4.6.3.5.2, 4.6.3.5.3, 4.6.4.2.3 a), 4.6.4.2.3 b), 4.6.4.2.3 c), and 4.8.4.3.3.1 of reference [1]) (NotifyInvocation) with eventname set to ‘xxx’ and event-value set to ‘yyy’. In case a notification does not use an event-value, ‘yyy’ shall be set to ‘empty’.* |
| *‘discard data unit’* | *4.8.3.3.2.5 of reference [1]*. |
| *‘discard data units in processing’* | *4.6.3.3.6 of reference [1]* |
| *‘procedure to association abort ‘xxx’’* | *(4.2.2.3 and 4.2.2.5 of reference [1]*) *raise ‘procedure to association abort ‘xxx’’ event with diagnostic set to ‘xxx’ to the Association Control procedure* |
| *‘terminate itself’* | *4.6.3.7 of reference [1]* |
| *‘wait’ <event>* | *(4.8.3.8 c) and 4.8.3.8 d) of reference [1]), wait until the event <event> occurs* |
| *‘set the data-unit-id parameter as per the received directivequalifier parameter*  *value’* | *4.8.3.8 e) of reference [1]* |
| ‘change the input queue size’ | 4.5.2 |
| ‘increment data units received’ | 4.5.3.4 a) |
| ‘increment data units submitted to processing’ | 4.5.4 a) |
| ‘increment current input queue’ | 4.5.3.4 b) |
| ‘decrement current input queue’ | 4.5.3.5, 4.5.4 b) |

**Table 4-9: Procedure State Table Compound Action Definitions**

|  |  |
| --- | --- |
| **Name** | **Actions Performed** |
| *{initiate stop}* | *‘clear the input queue’*  *‘complete data processing’*  *(+StopReturn)* |
| *{procedure to association abort ‘xxx’}* | *‘clear the input queue’*  *‘discard data units in processing’*  *‘procedure to association abort ‘xxx’’* |
| *{reset}* | *‘clear the Input Queue’*  *‘wait ‘data unit processing completed’’*  *‘wait ‘production status operational’’*  *‘set the data-unit-id parameter as per the received directive-qualifier parameter value’* |

## 5 BUFFERED FRAME DATA PROCESSING PROCEDURE

### 5.1 DISCUSSION

#### 5.1.1 PURPOSE

The BFDP procedure of the Forward Frame service is used to transfer SDLP Transfer Frames of one VC from the service user to the service provider.

The principal operational philosophy of the BFDP procedure is that the sequencing and timing of data contained within the Transfer Frames is ultimately maintained by protocols above the space data link layer (that is, above the layer at which the frames exist). Therefore the responsibility of the service performing the BFDP procedure is to allow the frames to flow into the forward link symbol stream with no inhibition on flow other than that possibly resulting from multiplexing with frames of other VCs.

#### 5.1.2 CONCEPT

The concept of the BFDP procedure is the same as that of the parent Buffered Data Processing procedure defined in reference [1], with additional data validation criteria that result in rejection of invalid PROCESS-DATA invocations and rejection notifications, and the addition of START invocation parameters.

data validation. The BFDP procedure validates the length and first four octets of the data parameter of the PROCESS-DATA invocation to ensure that it is carrying a valid Transfer Frame for the specific VC and SDLP being supported by the Forward Frame service instance to which the BFDP procedure instance belongs.

If the length of the frame in the data parameter is outside the range of the minimum and maximum permitted lengths, the PROCESS-DATA invocation is rejected and a NOTIFY invocation is sent to the service user identifying the rejected invocation and the reason for rejection.

For the purposes of the BFDP procedure, the pertinent Transfer Frame identification information comprises the TFVN (which identifies the SDLP), SCID, and VCID fields, which collectively comprise the GVCID. The BFDP procedure is configured to accept the frames with one GVCID. If the GVCID fields of the data parameter do not match their configured values, the PROCESS-DATA invocation is discarded and a NOTIFY invocation is sent to the service user identifying the rejected invocation and the reason for rejection.

NOTE – Assurance that multiple instances of the BFDP procedure (i.e., that multiple instances of the Forward Frame service) are not configured to permit transfer of frames for the same VC into the same forward space link is the responsibility of Service Management.

Additional BFDP START invocation parameters. The BFDP procedure adds the inputqueue-size, maximum-forward-buffer-size, and processing latency

limit parameters to the START invocation to allow the user of the service instance that contains the BFDP procedure to change the values of the input-queue-size, maximum-

forward-buffer-size, and processing latency limit configuration parameters of the BFDP procedure instance, as part of activating the procedure instance.

Service management of transfer mode. Subsection 4.7.2.2 of reference [1] defers to the individual service the specification of the method by which the transfer mode of this procedure is to be determined. This procedure establishes the transfer mode as a procedure configuration parameter, which in turn is set by a service management parameter.

### 5.2 PROCEDURE TYPE IDENTIFIER

The procedure type identifier bufferedFrameDataProcessing, as specified in annex B, shall be used for this procedure.

### 5.3 REFINEMENT

The BFDP procedure shall refine the PROCESS-DATA operation of the Framework Buffered Data Processing procedure by setting the type of the data parameter to be an octet string.

### 5.4 EXTENSION

**5.4.1** The BFDP procedure shall add the input-queue-size, maximum-forwardbuffer-size, and processing-latency-limit parameters to the START invocation of the Framework Buffered Data Processing procedure to allow the service user to change the values of the input-queue-size, maximum-forward-buffer-size, and processing-latency-limit dynamically modifiable configuration parameters as part of the activation of the BFDP procedure instance.

**5.4.2** The BFDP procedure shall extend the behavior of the Framework Buffered Data

Processing procedure by modifying the behavior of the procedure to

1. permit the modification of the input-queue-size, maximum-forwardbuffer-size, and processing-latency-limit configuration parameters as part of the activation of the procedure instance;
2. discard PROCESS-DATA invocations whose data parameter contents (1) do not match the GVCID configured for that Forward Frame service instance, (2) are shorter than the configured minimum length for that Forward Frame service, or (3) are longer than the configured maximum length for that Forward Frame service; and to send corresponding notifications to the user; and
3. initialize and update parameters that count the number of data units received, the number of data units submitted to processing, and the current size of the Input Queue.

### 5.5 BEHAVIOR

**5.5.1** The Notifications, Stopping, Terminating, and Aborting behaviors of the BFDP procedure shall be the same as those of the Buffered Data Processing procedure specified in 4.7.3.5, 4.7.3.6, 4.7.3.7, and 4.7.3.8, respectively, of reference [1].

**5.5.2** The Starting behavior of the BFDP procedure shall be the same as that of the Buffered Data Processing procedure, as specified in 4.7.3.1 of reference [1], with the addition of the following requirements:

**5.5.2.1** If the value of the input-queue-size parameter of the START invocation is ‘modify’, the value of the input-queue-size procedure configuration parameter of the BFDP procedure shall be set to the value of the input-queue-size parameter of the START invocation prior to the activation of the procedure.

**5.5.2.2** If the value of the processing-latency-limit parameter of the START invocation is ‘modify’, the value of the processing-latency-limit procedure configuration parameter of the BFDP procedure shall be set to the value of the processing-latency-limit parameter of the START invocation as part of the activation of the procedure.

**5.5.2.3** If the value of the maximum-forward-buffer-size parameter of the START invocation is ‘modify’, the value of the maximum-forward-buffer-size procedure configuration parameter of the BFDP procedure shall be set to the value of the maximum-forward-buffer-size parameter of the START invocation, as part of the activation of the procedure.

**5.5.3** The Transfer and Queuing of PROCESS-DATA Invocations behavior of the BFDP procedure shall be the same as that of the Buffered Data Processing procedure, as specified in 4.7.3.2 of reference [1], with the addition of the following requirements:

**5.5.3.1** If the result of the ANDing of the first 4 octets of the data parameter of the PROCESS-DATA invocation with the gvcid-bit-mask configuration parameter does not match the authorized-gvcid configuration parameter, the service provider shall invalidate the PROCESS-DATA invocation and invoke the NOTIFY operation with event-name set to ‘invalid GVCID’ and event-value set to the value of the dataunit-id parameter of the invalid PROCESS-DATA invocation.

1. For Telecommand frames, the value of the gvcid-bit-mask configuration parameter shall be

11000011 11111111 11111100 00000000 (C3 FF FC 00 Hex).

1. For Telecommand frames, the bits of the authorized-gvcid configuration parameter that correspond to the TFVN, SCID, and VCID fields of the Telecommand Transfer Frame Primary Header shall be set to the GCVID of the permitted virtual channel.
2. For AOS frames, the value of the gvcid-bit-mask parameter shall be 11111111 11111111 00000000 00000000 (FF FF 00 00 Hex).
3. For AOS frames, the bits of the authorized-gvcid configuration parameter that correspond to the TFVN, SCID, and VCID fields of the AOS Transfer Frame Primary Header shall be set to the GCVID of the permitted virtual channel.
4. For USLP frames, the value of the gvcid-bit-mask parameter shall be 11111111 11111111 11111111 11100000 (FF FF FF E0 Hex).
5. For USLP frames, the bits of the authorized-gvcid configuration parameter that correspond to the TFVN, SCID, Source or Destination ID, and VCID fields of the USLP Transfer Frame Primary Header shall be set to the GCVID of the permitted virtual channel.

NOTE – Frames transferred by the Forward Frame service contain the SCID of the destination space node. Therefore the Source or Destination ID bit of the authorized-gvcid configuration parameter that corresponds to the Source or Destination ID bit of the USLP Transfer Frame Primary Header (the 21st bit) shall be set to 1 when the Forward Frame service is provided by an ESLT.

**5.5.3.2** If the length of the data parameter of the PROCESS-DATA invocation is less than that specified by the minimum-frame-length configuration parameter, the service provider shall invalidate the PROCESS-DATA invocation and shall invoke the NOTIFY operation with event-name set to ‘frame too short’ and event-value set to the value of the data-unit-id parameter of the invalid PROCESS-DATA invocation.

**5.5.3.3** If the length of the data parameter of the PROCESS-DATA invocation is greater than that specified by the maximum-frame-length configuration parameter, the service provider shall invalidate the PROCESS-DATA invocation and invoke the NOTIFY operation with event-name set to ‘frame too long’ and event-value set to the value of the data-unit-id parameter of the invalid PROCESS-DATA invocation.

**5.5.3.4** Upon successful validation of the PROCESS-DATA invocation, the service

provider shall

1. place the PROCESS-DATA invocation in the Input Queue;
2. increment the value of the number-of-data-units-received parameter by

1; and

1. increment the input-queue-size parameter by 1.

**5.5.3.5** For each PROCESS-DATA invocation that is discarded from the Input Queue without being transferred to production processing, the service provider shall decrement the input-queue-size parameter by 1.

**5.5.4** The Processing of Data Units behavior of the BFDP procedure shall be the same as that of the Buffered Data Processing procedure, as specified in 4.7.3.3 of reference [1], with the addition of the following requirements:

Upon transfer of the data unit from the Input Queue to production processing, the service provider shall

1. increment the value of the number-of-data-units-submitted-to-processing parameter by 1; and
2. decrement the input-queue-size parameter by 1.

**5.5.5** The Positive Feedback requirements of the Buffered Data Processing procedure, as specified in 4.7.3.4 of reference [1], shall be replaced with the following requirement:

Upon the receipt of a ‘data unit processing completed’ event from service production, the service provider shall

1. increment the value of the number-of-data-units-processed parameter by 1; and
2. if the value of the process-completion-report parameter (see 4.6.4.1.4 of reference [1]) in the associated PROCESS-DATA invocation is ‘produce report’, send a NOTIFY invocation with Event Identifier ‘data unit processing completed’.

NOTE – The ‘data unit processing completed’ event is emitted by service production at the detected or estimated time of radiation of the data unit (see G2.3.4).

#### 5.5.6 INITIALIZATION OF DATA UNIT COUNTERS

At the start of the service instance provision period, the service provider shall initialize the values of the number-of-data-units-received, number-of-data-unitssubmitted-to-processing, and number-of-data-units-processed parameters to ‘zero’.

### 5.6 REQUIRED OPERATIONS

**5.6.1** The BFDP procedure shall use the STOP operation of the Buffered Data Processing procedure, as specified in reference [1], without extension or refinement.

**5.6.2** The BFDP procedure shall extend the START operation of the Buffered Data Processing procedure, as specified in 5.6.5.

**5.6.3** The BFDP procedure shall refine the PROCESS-DATA operation of the Buffered Data Processing procedure, as specified in 5.6.6.

**5.6.4** The BFDP procedure shall extend the NOTIFY operation of the Buffered Data Processing procedure, as specified in 5.6.7.

NOTE – Table 5-1 summarizes the operations of the BFDP procedure of the Forward Frame service.

#### Table 5-1: Buffered Frame Data Processing Procedure Required Operations

|  |  |  |  |
| --- | --- | --- | --- |
| **Operations** | **Extended** | **Refined** | **Procedure**  **Blocking/Non-Blocking** |
| START | Y | N | Blocking |
| STOP | N | N | Blocking |
| PROCESS-DATA | N | Y | Non-Blocking |
| NOTIFY | Y | N | Non-Blocking |

#### 5.6.5 START (CONFIRMED)

##### 5.6.5.1 Invocation, Return, and Parameters

**5.6.5.1.1** In addition to the parameters of the START operation of the Buffered Data Processing procedure as defined in 4.7.4.1 of reference [1], the extension parameter defined in table 5-2 shall be present in the START invocation of the BFDP procedure.

#### Table 5-2: START Extension Parameter

|  |  |  |
| --- | --- | --- |
| **Extension Parameter** | **Invocation** | **Return** |
| input-queue-size | M |  |
| maximum-forward-buffer-size | M |  |
| processing-latency-limit | M |  |

##### 5.6.5.1.2 Extension Parameter Syntax

The type BuffFrameDataProcStartInvocExt, as defined in annex D, shall define the syntax of the extension parameter of the START invocation.

##### 5.6.5.1.3 input-queue-size

The input-queue-size parameter shall have one of two values:

1. ‘unchanged’: The value of the input-queue-size configuration parameter of the BFDP procedure instance is to be unchanged from its current value.
2. ‘modify’: The value of the input-queue-size configuration parameter of the BFDP procedure instance is to be modified. The ‘modify’ value contains the value (in number of PROCESS-DATA invocations that can be stored in the queue) to which the input-queue-size configuration parameter is to be set.

**5.6.5.1.4** The input-queue-size configuration parameter shall be present in every START invocation.

##### 5.6.5.1.5 maximum-forward-buffer-size

The maximum-forward-buffer-size parameter shall have one of two values:

1. ‘unchanged’: The value of the maximum-forward-buffer-size configuration parameter of the BFDP procedure instance is to be unchanged from its current value.
2. ‘modify’: The value of the maximum-forward-buffer-size configuration parameter of the BFDP procedure instance is to be modified. The ‘modify’ value contains the value (in number of PROCESS-DATA invocations that can be stored in the forward buffer) to which the maximum-forward-buffer-size configuration parameter is to be set.

**5.6.5.1.6** The maximum-forward-buffer-size configuration parameter shall be present in every START invocation.

##### 5.6.5.1.7 processing-latency-limit

The processing-latency-limit parameter shall have one of two values:

1. ‘unchanged’: The value of the processing-latency-limit configuration parameter of the BFDP procedure instance is to be unchanged from its current value.
2. ‘modify’: The value of the processing-latency-limit configuration parameter of the BFDP procedure instance is to be modified. The ‘modify’ value contains the value (in milliseconds) to which the processing-latency-limit configuration parameter is to be set.

**5.6.5.1.8** The processing-latency-limit configuration parameter shall be present in every START invocation.

#### 5.6.6 PROCESS-DATA (UNCONFIRMED)—INVOCATION PARAMETER—DATA PARAMETER REFINEMENT

The data parameter of the PROCESS-DATA invocation shall be formatted as an octet string.

NOTE – The data parameter is expected to contain a fixed-length Transfer Frame for either the AOS or Unified space data link protocol, or a CADU containing a (possibly encoded) CCSDS-standard fixed-length Transfer Frame. However, the Forward Frame service will actually validate any octet string with any format as long as (a) the first four octets of the octet string, when ANDed with the gvcid-

bit-mask configuration parameter, matches the authorized-gvcid configuration parameter; and (b) the length of the octet string is within the configured minimum and maximum lengths. Such a data unit would of course have to have corresponding production processing functions associated with it, and Service Management would have to be able to configure the Forward Frame service instance to operate correctly with those production processing functions.

#### 5.6.7 NOTIFY (UNCONFIRMED)

##### 5.6.7.1 General

The BFDP procedure shall extend the NOTIFY operation of the Buffered Data Processing procedure defined in 4.7.4.2 of reference [1] through the definition of three additional possible events (event names with their event values).

##### 5.6.7.2 Invocation and Parameters

###### 5.6.7.2.1 event-name Extension

**5.6.7.2.1.1** The value of the event-name parameter shall be one of the following:

1. one of the values specified for the NOTIFY operation of the Data Processing procedure in 4.6.4.2.3 of reference [1];
2. ‘invalid GVCID’ (event-name)—in the data parameter of a PROCESS-DATA invocation, the header fields corresponding to those of the GVCID do not match the authorized GVCID for this procedure instance (see 4.5.3.1):
   1. the event-name of this event shall contain the procedure instance identifier of the BFDP procedure instance triggering the event;
   2. the associated event-value shall contain the value of the data-unit-id of the failed PROCESS-DATA invocation, where the path specifying the type to be used is defined in 5.6.7.2.1.2;
3. ‘frame too short’ (event-name)—in the data parameter of a PROCESS-DATA invocation, the length of the data parameter of the PROCESS-DATA invocation is less than the minimum frame length specified for this procedure instance (see 4.5.3.2):
   1. the event-name of this event shall contain the procedure instance identifier of the BFDP procedure instance triggering the event;
   2. the associated event-value shall contain the value of the data-unit-id of the failed PROCESS-DATA invocation, where the path specifying the type to be used is defined in 5.6.7.2.1.2;
4. ‘frame too long’ (event-name)—in the data parameter of a PROCESS-DATA invocation, the length of the data parameter of the PROCESS-DATA invocation is greater than the maximum frame length specified for this procedure instance (see 4.5.3.3):
   1. the event-name of this event shall contain the procedure instance identifier of the BFDP procedure instance triggering the event;
   2. the associated event-value shall contain the value of the data-unit-id of the failed PROCESS-DATA invocation; the path specifying the type to be used is defined in 5.6.7.2.1.2.

**5.6.7.2.1.2** The data type of the event-value parameter for the ‘invalid GVCID’, ‘frame too short’, and ‘frame too long’ events uses the following path:

1. the first part of the path is ‘NotifyInvocation’: ‘eventValue’: ‘EventValue’: ‘qualifiedValue’: ‘SequenceOfQualifiedValue’: ‘SEQUENCE OF QualifiedValue’, where this sequence has the length 1;
2. the second part of the path is ‘QualifiedValue’: ‘valid’: ‘TypeAndValueComplexQualified’: ‘typeAndValue’: ‘Embedded’: ‘EMBEDDED PDV’, where the Object Identifier and the type of the data-unit-id parameter are pBFDPdataUnitId asspecified in annex D and DataUnitId as specified in F3.3 of reference [1], respectively.

NOTE – All relevant types are defined in F3.3 of reference [1].

**5.6.7.2.1.3** The Published Identifier (i.e., Event Identifier) for the event-name of the ‘invalid GVCID’ event is specified in annex F as pBFDPinvalidGvcid.

**5.6.7.2.1.4** The Published Identifier (i.e., Event Identifier) for the event-name of the ‘frame too short’ event is specified in annex F as pBFDPframeTooShort.

**5.6.7.2.1.5** The Published Identifier (i.e., Event Identifier) for the event-name of the ‘frame too long’ event is specified in annex F as pBFDPframeTooLong.

### 5.7 CONFIGURATION PARAMETERS

**5.7.1** The BFDP procedure configuration parameters that need to be configured in the context of the procedure shall be as defined in table 5-3.

NOTES

1. For each configuration parameter, the table provides the engineering unit (if applicable) a cross reference to the use of the parameter in the specification of the procedure, identifies whether the parameter may be read and/or dynamically modified, and also identifies the Parameter Identifier and type to be used in reporting the value of the parameter.
2. Section 1.6.1.7.37 of reference [1] states, “Unless explicitly specified otherwise in the Framework definition of a procedure type, the method(s) by which the values of dynamically modifiable procedure configuration parameters are modified is (are) delegated to the service using the procedure or to a derived procedure.” For the Buffered Frame Data Processing procedure, the only dynamically modifiable configuration parameters are the input-queue-size, maximum-forwardbuffer-size, and processing-latency-limit parameters. For the Forward Frame service, the only method by which the values of these parameters can be modified is through the START invocation as specified in 5.6.5.1.3.

#### Table 5-3: Buffered Frame Data Processing Procedure Configuration Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameters** | **Cross-**  **Reference** | **Readable** | **Dynam-**  **ically modifiable** | **Configuration Parameter Identifier and Type** |
| input-queue-size  (in number of  PROCESS-DATA  invocations the queue will store) | CSTS SFW  (reference [1]),  4.6.3.2.3 | Yes | Yes | pDPinputQueueSize PDPinputQueueSizeType  (CSTS SFW F3.16) |
| maximum-forwardbuffer-size (in number of  PROCESS-DATA  invocations the buffer will store) | CSTS SFW  (reference [1]),  4.7.3.2.1.2 | Yes | Yes | pBDPmaxForwardBufferSize PBDPmaxForwardBufferSizeType  (CSTS SFW F3.16) |
| processinglatency-limit (in milliseconds) | CSTS SFW  (reference [1]),  4.7.3.2.2.7 | Yes | Yes | pBDPprocessingLatencyLimit PBDPprocessingLatencyLimitType (CSTS SFW F3.16) |
| data-transfermode (complete, timely) | CSTS SFW  (reference [1]),  4.7.3.2.2 | Yes | No | pBDPdataTransferMode  PBDPdataTransferModeType (CSTS SFW F3.16) ) |
| gvcid-bit-mask  (4-octet) | 5.5.3.1 | Yes | No | pBFDPgvcidBitMask PBFDPgvcidBitMaskType  (annex F) |
| authorized-gvcid  (4-octet) | 5.5.3.1 | Yes | No | pBFDPauthorizedGvcid PBFDPauthorizedGvcidType  (annex F) |
| minimum-frame  length (in octets) | 5.5.3.2 | Yes | No | pBFDPminFrameLen PBFDPminFrameLenType  (annex F) |
| maximum-frame  length (in octets) | 5.5.3.3 | Yes | No | pBFDPmaxFrameLen PBFDPmaxFrameLenType  (annex F) |

**5.7.2** The data-transfer-mode shall be set to the value of the bufferedData:dataTransferMode component of the service management parameter with the classifier ffDataProcessingMode.

**5.7.3** The gvcid-bit-mask shall be set to the bit mask pattern for the applicable space data link protocol data unit type (TC, AOS, USLP, CADU, or other) that is specified in the service management parameter with the classifier ffAuthorizedGvcidAndBitMask.

**5.7.4** The authorized-gvcid shall be set to the value that is specified in the service management parameter with the classifier ffAuthorizedGvcidAndBitMask.

**5.7.5** The minimum-frame-length shall be set to the value of the service management parameter with the classifier ffMinFrameLength.

**5.7.6** The maximum-frame-length shall be set to the value of the service management parameter with the classifier ffMaxFrameLength.

NOTES

1. The classifiers for the input-queue-size, maximum-forward-buffersize, and processing-latency-limit configuration parameters are specified in 7.4.1, 7.4.2, and 7.4.3, respectively.
2. If the procedure is used to transfer fixed-length frames, the values of the ffMinFrameLength and ffMaxFrameLength parameters are the same.
3. The Object Identifiers associated with the service management parameter classifiers ffDataTransferMode, ffAuthorizedGvcidAndBitMask, ffMinFrameLength, and ffMaxFrameLength are assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 1} subtree (see annex B).

### 5.8 READ-ONLY PARAMETERS

**5.8.1** The BFDP procedure read-only parameters shall be as defined in table 5-4.

NOTE – For each read-only parameter, the table provides the engineering unit (if applicable), a cross reference to the use of the parameter in the specification of the procedure, the classifier of the parameter, the type to be used in reporting the value of the parameter, and the annex in which that type is specified.

#### Table 5-4: Buffered Frame Data Processing Procedure Read-Only Parameters

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Cross-Reference** | **Read-Only Parameter Classifier and Type** |
| number-of-dataunits-received (in number of  PROCESS-DATA  invocations) | 5.5.3.4 b), 5.5.6 | pBFDPnumberDataUnitsRcvd PBFDPnumberDataUnitsRcvdType  (annex F) |
| number-of-dataunits-submittedto-processing (in number of  PROCESS-DATA  invocations) | 5.5.4 a), 5.5.6 | pBFDPnumberDataUnitsToProcessing PBFDPnumberDataUnitsToProcessingType  (annex F) |
| number-of-dataunits-processed  (in number of  PROCESS-DATA  invocations) | 5.5.5 a), 5.5.6 | pBFDPnumberDataUnitsProcessed PBFDPenumberDataUnitsProcessedType (annex F) |

**5.8.2** When the Buffered Frame Data Processing Procedure is the prime procedure type, the functional resource parameter (see 1.6.1.12.3) with the classifier ffNumberDataUnitsRcvd shall be set to the value of the BFDP number-of-dataunits-received parameter.

**5.8.3** When the Buffered Frame Data Processing Procedure is the prime procedure type, the functional resource parameter with the classifier ffNumberDataUnitsToProcessing shall be set to the value of the BFDP number-of-data-units-submitted-toprocessing parameter.

**5.8.4** When the Buffered Frame Data Processing Procedure is the prime procedure type, the functional resource parameter with the classifier ffNumberDataUnitsProcessed shall be set to the value of the BFDP number-of-data-units-processed parameter.

### 5.9 PROCEDURE STATE TABLE

NOTE – In the following tables, the specifications that are inherited from the parent Buffered Data Processing procedure are *italicized*, and the additions to the tables that are specific to the BFDP procedure are in plain text.

#### Table 5-5: Buffered Frame Data Processing Procedure State Table

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Incoming Event** | **State 1**  **(‘inactive’)** | **State 2 (‘active’)** |
| 1 | *(StartInvocation)* | *IF “positive result”*  *THEN*  {change modifiable parameters} set “reading suspended” to  FALSE  *(+StartReturn)*  *→ 2*  *ELSE*  *(-StartReturn)*  *ENDIF* | *{procedure to association abort ‘protocol error’}*  →*1* |
| 2 | *(StopInvocation)* | *{procedure to association abort*  *‘protocol error’}* | *IF “positive result”*  *THEN*  *{initiate stop}*  →*1 ELSE*  *(-StopReturn)*  *ENDIF* |
| 3 | *(ProcessDataInvocation)[[1]](#footnote-1)* | *{procedure to association abort*  *‘protocol error’}* | IF “valid invocation”  THEN  *IF “timely mode”*  *THEN*  *IF “queue overflow”*  *THEN*  *‘discard oldest data units’*  *ENDIF*  *ENDIF*  *‘queue data unit’*  ‘increment data units received’  ‘increment current input queue’  *IF “complete mode” THEN*  *IF “queue full”*  *THEN*  *‘suspend reading’*  *set “reading suspended” to TRUE*  *ENDIF*  *ENDIF*  ELSE  ‘notify invalid invocation’  ENDIF |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Incoming Event** | | **State 1**  **(‘inactive’)** | **State 2 (‘active’)** | |
| 4 | *(ForwardBuffer)* |  | *{procedure to association abort*  *‘protocol error’}* | FOR\_EACH (ProcessDataInvocation)  IN (ForwardBuffer)  IF “valid invocation”  THEN  *IF “timely mode”*  *THEN*  *IF “queue overflow”*  *THEN*  *‘discard oldest data units’*  *ENDIF*  *ENDIF*  *‘queue data unit’*  ‘increment data units received’  ‘increment current input queue’  *IF “complete mode”*  *THEN*  *IF “queue full”*  *THEN*  *‘suspend reading’*  *set “reading suspended”*  *TRUE*  *ENDIF*  *ENDIF*  ELSE  ‘notify invalid invocation’  ENDIF  ENDFOR\_EACH | *to* |
| 5 | *‘data unit ready’* |  | *Not applicable* | *‘process data unit’*  ‘increment data units submitted to processing’  ‘decrement current input queue’ |  |
| 6 | *‘data unit processing*  *completed’* | | *[ignore]* | *IF “report”*  *THEN*  *‘notify ‘data processing completed’/*  *‘empty’’*  *ENDIF*  *IF “reading suspended”*  *THEN*  *IF (NOT “queue full”)*  *THEN*  *‘resume reading’*  *set “reading suspended” to FALSE*  *ENDIF*  *ENDIF* |  |
| 7 | *‘processing latency timer*  *expired’* | | *Not applicable* | *‘discard data unit’* | |
| 8 | *‘production status change to*  *‘interrupted’’* | | *[ignore]* | *‘discard data units in processing’*  *‘notify ‘production status change’ / ‘interrupted’’* | |
| 9 | *‘production status change to*  *‘halted’’* | | *[ignore]* | *‘discard data units in processing’*  *‘notify ‘production status change’ / ‘halted’’* | |
| 10 | *‘production status change to*  *‘operational’’* | | *[ignore]* | *‘notify ‘production status change’ /*  *‘operational’’* | |
| 11 | *‘production status change to*  *‘configured’’* | | *[ignore]* | *‘notify ‘production status change’ / ‘configured’’* | |
| 12 | *‘production configuration change’* | | *[ignore]* | *‘notify ‘production configuration change’ / ‘empty’’* | |
| **No.** | **Incoming Event** | | **State 1**  **(‘inactive’)** | **State 2 (‘active’)** | |
| 13 | *‘data processing configuration change’* | | *[ignore]* | *‘notify ‘data processing configuration change’ / ‘procedure configuration parameter values’’ (see 4.7.4.2.2.2 of reference [1])* | |
| 14 | *‘invalid PDU ‘xxx’’* | | *{procedure to association abort*  *‘xxx’}* | *{procedure to association abort ‘xxx’}*  →*1* | |
| 15 | *‘terminate procedure’* | | *‘terminate itself’* | *‘terminate itself’)* | |

#### Table 5-6: Procedure State Table Incoming Event Description References

|  |  |
| --- | --- |
| **Event** | **Reference** |
| *‘data unit processing completed’* | 5.5.5 |
| *‘data unit ready’* | *4.6.3.3.1 of reference [1]* |
| *‘processing latency timer expired’* | *4.7.3.2.2.11 of reference [1]* |
| *‘terminate procedure’* | *4.2.3 of reference [1]* |
| *‘production status change*  *to ‘xxx’’* | *B2.5 of reference [1]* |
| *‘production configuration change’* | *3.11.2.2.3.2 b) of reference [1]* |
| *‘data processing configuration change’* | *4.7.4.2.2.2 of reference [1]* |
| *‘invalid PDU ‘xxx’’* | *(3.2.3.6, 4.2.2.4, 4.7.3.2.1.4 of reference [1]), where ‘xxx’ is one of the diagnostic values specified in (4.2.2.5 or 4.7.3.2.1.4 of reference [1]).* |

#### Table 5-7: Procedure State Table Predicate Descriptions

|  |  |
| --- | --- |
| **Predicate** | **Evaluates to TRUE if** |
| *“queue overflow”* | *There is not sufficient space on the Input Queue to store the PROCESS-DATA invocations received (see 4.7.3.2.2.4 of reference [1]): the transfer mode is ‘timely’.* |
| *“queue full”* | *There is not enough space on the Input Queue to store all PROCESS-DATA invocations that might be contained in a maximum sized forward buffer (see 4.7.3.2.2.3 of reference [1]): the transfer mode is ‘complete’.* |
| *“complete mode”* | *Complete transfer mode is in effect, based on the setting of the data-transfer-mode procedure configuration parameter*  *(table* 5-3*).* |
| **Predicate** | **Evaluates to TRUE if** |
| *“timely mode”* | *Timely transfer mode is in effect, based on the setting of the datatransfer-mode procedure configuration parameter (table* 5-3*).* |
| *“positive result”* | *No reason for sending a negative return has been detected; that is, for the START invocation, none of the conditions in 3.7.2.3.1 or 4.6.3.1 a) of reference [1] applies, and for the STOP invocation, none of the conditions in 3.3.2.7.1 of reference [1] applies.* |
| *“report”* | *The process-completion-report parameter value in the associated ProcessDataInvocation PDU is ‘produce report’ (see 4.6.4.1.4 of reference [1]).* |
| *“valid invocation”* | *A*ll validity checks specified in 5.5.3.1, 5.5.3.2, and 5.5.3.3 are passed. |

#### Table 5-8: Procedure State Table Boolean Flags

|  |  |
| --- | --- |
| **Predicate** | **Set to TRUE if** |
| *“reading suspended”* | *In complete transfer mode reading data from the data communications service has been suspended.* |

#### Table 5-9: Procedure State Table Simple Action References

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | **References** | | |
| *‘queue data unit’* | | *4.7.3.2 of reference [1]* | | |
| *‘complete data processing’* | | *4.6.3.6.2 b) of reference [1]* | | |
| *‘clear the Input Queue’* | | *4.6.3.6.2 a) of reference [1].* | | |
| *‘suspend reading’* | | *4.7.3.2 of reference [1]* | | |
| *‘resume reading’* | | *4.7.3.2 of reference [1]* | | |
| *‘discard oldest data units’* | | *4.7.3.2 of reference [1]* | | |
| *‘discard data unit’* | | *(4.7.3.2.2.11 of reference [1]),* 5.5.3.1, 5.5.3.2, 5.5.3.3 | | |
| *‘discard data units in processing’* | | *4.6.3.3.6 of reference [1]* | | |
| *‘notify ‘xxx’ / ‘yyy’’* | | *(NotifyInvocation) with event-name set to ‘xxx’ and event-value set to ‘yyy’. (4.6.3.5.1.1, 4.6.3.5.2, 4.6.3.5.3, 4.6.4.2.3 a), 4.6.4.2.3 b), 4.6.4.2.3 c), and 4.7.4.2.2.2 of reference [1])*. | | |
| ‘notify invalid invocation’ | | Send a (NotifyInvocation) with event-name set to the event as defined in 5.5.3.1, 5.5.3.2, or 5.5.3.3 (as appropriate), and eventvalue set to the value of the data-unit-id parameter of the invalid invocation. | | |
| *‘procedure to association abort ‘xxx’’* | | *(4.2.2.3 and 4.2.2.5 of reference [1]) raise ‘procedure to association abort ‘xxx’’ event with diagnostic set to ‘xxx’ to the Association Control procedure* | | |
| **Name** | |  | **References** | |
| *‘terminate itself’* | | *4.6.3.7 of reference [1]* |  | |
| ‘change the input queue size’ | | 5.5.2.1 |  | |
| ‘change the maximum forward buffer size’ | | 5.5.2.3 |  | |
| ‘change the processing latency limit’ | | 5.5.2.2 |  | |
| ‘increment data units received’ | | 5.5.3.4 b) |  | |
| ‘increment data units submitted to processing’ | | 5.5.4 a) |  | |
| ‘increment current input queue’ | | 5.5.3.4 c) |  | |
| ‘decrement current input queue’ | | 5.5.3.5, 5.5.4 b) |  | |

**Table 5-10: Procedure State Table Compound Action Definitions**

|  |  |
| --- | --- |
| **Name** | **Actions Performed** |
| *{procedure to association abort ‘xxx’}* | *‘clear the input queue’*  *‘discard data units in processing’*  *‘procedure to association abort ‘xxx’’* |
| *{initiate stop}* | *‘clear the input queue’*  *‘complete data processing’*  *(+StopReturn)* |
| {change modifiable parameters} | ‘change the input queue size’  ‘change the maximum forward buffer size’  ‘change the processing latency limit’ |

## 6 MASTER THROW EVENT PROCEDURE

### 6.1 DISCUSSION

#### 6.1.1 PURPOSE

The Master Throw Event procedure of the Forward Frame service is used to allow a service user to reconfigure the values of dynamically modifiable configuration parameters of the FR instances that comprise the production processes that directly support that instance of the Forward Frame service.

#### 6.1.2 CONCEPT

Many Provider CSSSes extend the capability to dynamically modify some configuration parameters (e.g., bit rate, modulation index) of the production processes that support cross transfer services such as the Forward Frame service. The Master Throw Event procedure allows a user of a Forward Frame service instance to be able to change the modifiable configuration parameters of the FR instances that support that specific service instance.

NOTE – The prospective CCSDS SC-CSTS will allow the user of that service to change the modifiable configuration parameters of *all* FR instances in a Service Package. Once the SC-CSTS becomes available, the Master Throw Event procedure of the Forward Frame service is expected to be deprecated.

Because the same set of FR instances can be shared by multiple Forward Frame service instances (as well as other higher-layer protocols, such as those for Space Packets and Space Internetworking), operational policies normally mandate that only one service instance per forward space link be given control over the configuration of those shared resources. This service instance is referred to herein as the *master service instance*, and the user of that service instance is referred to as the *master user*.

To support the enforcement of the operational policy of only one master instance/user per shared forward space link, the Master Throw Event procedure has an enable-masterthrow-event procedure configuration parameter that controls whether or not the Master Throw Event procedure of a given Forward Frame service can be activated. The enablemaster-throw-event procedure configuration parameter is in turn configured by a service management parameter.

NOTE – It is ultimately the responsibility of Service Management to enforce the policy by configuring only one instance of Forward Frame service per forward space link to be enabled to use the Master Throw Event procedure.

Procedure enablement validation. The Master Throw Event procedure validates any EXECUTE-DIRECTIVE invocation with procedure type identifier masterThrowEvent against the value of the enable-master-throw-event procedure configuration parameter for that procedure instance. If the value is ‘enabled’, then the EXECUTEDIRECTIVE is processed in accordance with the Throw Event procedure specified in reference [1]. If the value is ‘disabled’, the EXECUTE-DIRECTIVE invocation is rejected.

Page 6-1

### 6.2 PROCEDURE TYPE IDENTIFIER

The procedure type identifier masterThrowEvent, as specified in annex B, shall be used for this procedure.

### 6.3 REFINEMENT

The Master Throw Event procedure shall refine the behavior of the Throw Event procedure specified in reference [1] by limiting the valid directives to those of FR instances that are directly associated with the provision or production of the Forward Frame service instance that is executing the Master Throw Event procedure.

NOTE – The default scope of the EXECUTE-DIRECTIVE operation of the Framework Throw Event procedure allows it to transfer directives for (a) the Throw Event procedure instance itself, (b) any instance of other procedure of the service executing the Throw Event procedure and (c) any FR instance that is directly associated with the provision or production of the Forward Frame service instance that is executing the Throw Event procedure (see 3.13.1.1 of reference [1]). There are no directives defined for the Master Throw Event procedure, so case (a) is non-applicable. However, the Forward Frame service does include the Sequence-Controlled Frame Data Processing procedure, for which the ‘reset’ directive exists; the Master Throw Event procedure explicitly eliminates case (b) to disallow the invocation of the ‘reset’ directive through the Master Throw Event. Case (c) is the only valid case for the Master Throw Event procedure.

### 6.4 EXTENSION

**6.4.1** The Master Throw Event procedure shall extend the Framework Throw Event procedure by adding the enable-master-throw-event procedure configuration parameter to enable the use of the procedure.

**6.4.2** The Master Throw Event procedure shall add the ‘master throw event disabled’ diagnostic to the EXECUTE-DIRECTIVE negative result acknowledgement of the Framework Throw Event procedure to indicate that the attempt to use the procedure has been denied.

### 6.5 BEHAVIOR

**6.5.1** The behavior of the Master Throw Event procedure shall be the same as that of the Throw Event procedure (4.12 of reference [1]), with the following requirements:

**6.5.2** If the master-throw-event-enabled procedure configuration parameter has the value ‘disabled’, the service provider shall reject every EXECUTE-DIRECTIVE invocation and send a negative acknowledgement with diagnostic ‘master throw event disabled’.

Page 6-2

**6.5.3** In order to be valid, the directive must be for a FR instance that is directly associated with the Forward Frame service instance that is executing the Master Throw Event procedure.

**6.5.4** If the directive-identifier is for either the Master Throw Event procedure itself or for any instance of any other procedure of the Forward Frame service, the service provider shall reject the EXECUTE-DIRECTIVE invocation and send a negative acknowledgement with diagnostic ‘unknown directive’.

### 6.6 REQUIRED OPERATIONS

**6.6.1** The Master Throw Event procedure shall extend the EXECUTE-DIRECTIVE operation of the Throw Event procedure, as specified in 6.6.2.

NOTE – Table 6-1 summarizes the operations of the Master Throw Event procedure of the Forward Frame service.

#### Table 6-1: Master Throw Event Procedure Required Operation

|  |  |  |  |
| --- | --- | --- | --- |
| **Operations** | **Extended** | **Refined** | **Procedure**  **Blocking/Non-Blocking** |
| EXECUTE-  DIRECTIVE | Y | N | Non-Blocking |

#### 6.6.2 EXECUTE-DIRECTIVE (CONFIRMED)—INVOCATION, RETURN, AND PARAMETER—ACKNOWLEDGEMENT DIAGNOSTIC EXTENSION

**6.6.2.1** If an EXECUTE-DIRECTIVE negative acknowledgement is sent, the diagnostic parameter shall contain one of the following values:

1. one of the diagnostic values specified in 4.12.4.1.3.1 of reference [1]; or
2. ‘master throw event disabled’; the Master Throw Event procedure cannot be used by this service instance.

**6.6.2.2** The type MasterTeExecDirNegAckDiagnosticExt, as defined in annex E, shall specify the additional value of the diagnostic parameter of the EXECUTEDIRECTIVE negative acknowledgement.

Page 6-3

### 6.7 CONFIGURATION PARAMETERS

**6.7.1** The Master Throw Event procedure configuration parameters that need to be configured in the context of the procedure shall be as defined in table 6-2.

NOTE – For each configuration parameter, the table provides the engineering unit (if applicable), provides a cross reference to the use of the parameter in the specification of the procedure, identifies whether the parameter may be read and/or dynamically modified, and also identifies the Parameter Identifier and type to be used in reporting the value of the parameter.

#### Table 6-2: Master Throw Event Procedure Configuration Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameters** | **Cross-**  **Reference** | **Readable** | **Dynam**  **-ically modifiable** | **Configuration Parameter Identifier and Type** |
| master-throwevent-enabled | 6.5.2 | Yes | No | pMTEenableThrowEvent PMTEenableThrowEventType  (annex F) |

**6.7.2** The master-throw-event-enabled shall be set to the value of the service management parameter with the classifier ffMasterThrowEventEnabled.

NOTE – The Object Identifiers associated with the service management parameter classifier ffMasterThrowEventEnabled is assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 1} subtree (see annex B).

### 6.8 PROCEDURE STATE TABLE

**6.8.1** The Master Throw Event procedure adopts the Throw Event Procedure State Table, Procedure State Table Incoming Event Description References, and Procedure State Table Simple Action References tables (tables 4-71, 4-72, and 4-74, respectively, of reference [1]).

**6.8.2** The Master Throw Event procedure modifies the definition of the ‘valid directive’ predicate (Procedure State Table Predicate Definitions, table 4-73 of reference [1]) as follows: the predicate ‘valid directive’ evaluates to TRUE if ‘none of the error conditions identified in 4.12.4.1.3.1 of reference [1]), 6.5.2, or 6.5.4 are true for the EXECUTEDIRECTIVE invocation.’

Page 6-4

## 7 SETTING OF SERVICE MANAGEMENT AND CONFIGURATION PARAMETERS INHERITED FROM FRAMEWORK OPERATIONS AND PROCEDURES

### 7.1 GENERAL

The procedures specified in reference [1] define configuration parameters for the Framework procedures, but defer to the derived services the specification of the method by which each of those configuration parameters is to be set. Sections 7.2 through 7.7 specify the method by which each of the Framework procedure configuration parameters is to be set for the FF-CSTS.

For each of the procedure configuration parameters that are specified to be a service management parameter, the classifier for each parameter is also specified. The parameterId corresponding to each such classifier is defined in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 1} subtree (see annex B). The type of each service management parameter is the same as that of the corresponding procedure parameter.

### 7.2 ASSOCIATION CONTROL PROCEDURE CONFIGURATION PARAMETERS

**7.2.1** The initiator-identifier configuration parameter (4.3.5 of reference [1]) shall be set by the service management parameter with the classifier ffInitiatorId.

**7.2.2** The responder-identifier configuration parameter (4.3.5 of reference [1]) shall be set by the service management parameter with the classifier ffResponderId.

**7.2.3** The service-instance-identifier parameter (4.3.5 of reference [1]) shall be set by the service management parameter with the classifier ffSvcInstanceId.

**7.2.4** The responder-port-identifier service management parameter (3.4.2.2.4.3 and 4.3.5 of reference [1]) shall be set by the service management parameter with the classifier ffResponderPortId.

NOTE – As described in the specification of the responder-port-identifier parameter (specified in reference [1]), the contents of the parameter are not used by the Association Control procedure itself, but rather by the underlying communications service that delivers the incoming PDUs to CSTS provider. The purpose of assigning a classifier and parameterId to this parameter is to allow its value to be reported or queried.

### 7.3 SEQUENCE-CONTROLLED FRAME DATA PROCESSING PROCEDURE CONFIGURATION PARAMETERS

The input-queue-size configuration parameter (4.8.5 of reference [1]) shall be set by the service management parameter with the classifier ffInputQueueSize.

Page 7-1

### 7.4 BUFFERED FRAME DATA PROCESSING PROCEDURE CONFIGURATION PARAMETERS

**7.4.1** The input-queue-size configuration parameter (4.7.5 of reference [1]) shall be set by the service management parameter with the classifier ffInputQueueSize.

**7.4.2** The maximum-forward-buffer-size configuration parameter (4.7.5 of reference [1]) shall be set by the service management parameter with the classifier ffMaxFwdBufferSize.

**7.4.3** The processing-latency-limit configuration parameter (4.7.5 of reference [1]) shall be set by the service management parameter with the classifier ffProcedureLatencyLimit.

### 7.5 CYCLIC REPORT PROCEDURE CONFIGURATION PARAMETERS

NOTE – The Cyclic Report procedure is optional. If the Cyclic Report procedure is not available in an instance of the Forward Frame service, the values of the procedure configuration parameters associated with the Cyclic Report procedure are undefined.

**7.5.1** The set of named Parameter Label Lists that constitutes the named-label-lists configuration parameter (4.10.5 of reference [1]) shall be set by the service management parameter with the classifier ffNamedParamLabelLists.

**7.5.2** The set of named Parameter Label Lists shall be used by all instances of Cyclic Report and Information Query procedures of the FF-CSTS instance.

**7.5.3** Service Management shall designate, at most, one list in the set of Parameter Label Lists as the default list of parameters (4.10.5 of reference [1]).

**7.5.4** The default list of Parameter Labels shall be used as the default Parameter Label List by all instances of Cyclic Report and Information Query procedures of the FF-CSTS instance.

**7.5.5** The minimum-allowed-delivery-cycle configuration parameter (4.10.5 of reference [1]) of every instance of the Cyclic Report procedure in the FF-CSTS instance shall be set by the service management parameter with classifier ffMinAllowedDeliveryCycle.

Page 7-2

### 7.6 NOTIFICATION PROCEDURE CONFIGURATION PARAMETERS

NOTE – The Notification procedure is optional. If the Notification procedure is not available in an instance of the Forward Frame service, the values of the events associated with the Notification procedure are undefined.

**7.6.1** The set of named Event Label Lists that constitutes the named-label-lists configuration parameter (4.11.5 of reference [1]) shall be a service management parameter with the classifier ffNamedEventLabelLists.

**7.6.2** The set of named Event Label Lists shall be available for use by the Notification procedure instance of the Forward Frame transfer service instance.

**7.6.3** Service Management shall designate at most one list in the set of Event Label Lists as the default list of events (4.11.5 of reference [1]).

**7.6.4** The default list of events shall be used as the default Event Label List by the Notification procedure instance of the FF-CSTS instance.

### 7.7 INFORMATION QUERY PROCEDURE CONFIGURATION PARAMETERS

NOTE – The Information Query procedure is optional. If the Information Query procedure is not available in an instance of the Forward Frame service, the values of the parameters associated with the Information Query procedure are undefined.

The set of named Parameter Label Lists that constitutes the named label lists configuration parameter (4.9.5 of reference [1]) shall be a service management parameter with the classifier ffNamedParamLabelLists.

NOTE – Requirements on the composition of the set of named Parameter Label Lists with the classifier ffNamedParamLabelLists are specified in 7.5.2, 7.5.3, and 7.5.4.

### 7.8 MASTER THROW EVENT PROCEDURE CONFIGURATION PARAMETERS

Service Management specifies the configuration of the set of FR instances that are directly associated with the production of the Forward Frame service instance. The predefined actions and associated directive IDs, directive values, and guard conditions that are defined for that set of associated FR instances collectively form the set of those configuration parameter values that apply to the Forward Frame service instance.

Page 7-3

## 8 FORWARD FRAME SERVICE-LEVEL FUNCTIONAL RESOURCE PARAMETERS AND EVENTS

### 8.1 GENERAL

The CSTS Specification Framework (reference [1]) specifies that certain service-level parameters and events be specified for the functional resource that represents each CSTS provider. The required functional resource parameters are associated with the returntimeout and production-status parameters of the CSTS. The required functional resource events notify the occurrence of the ‘production status change’ and ‘production configuration change’ events of the CSTS.

### 8.2 ffResponseTimeout FUNCTIONAL RESOURCE PARAMETER

#### 8.2.1 GENERAL

The CSTS Specification Framework specifies that each CSTS define a responsetimeout functional resource configuration parameter (3.2.1.2 of reference [1]).

NOTE – As described in the specification of the response-timeout parameter in reference [1], the parameter is not used by any of the procedures of a CSTS provider unless that CSTS uses one or more provider-invoked procedures. The Forward Frame CSTS does not have any provider-invoked procedures, so the CSTS service provider may use the response-timeout parameter value to anticipate how long it (the service provider) has to respond to the invocation of any of the confirmed operations to avoid causing the service user to invoke PEER-ABORT.

**8.2.2** The response-timeout functional resource configuration parameter (3.2.1.2 of reference [1]) shall be configured and reported by the Forward Frame functional resource parameter with the classifier ffResponseTimeout.

**8.2.3** The Published Identifier associated with the parameter classifier ffResponseTimeout is assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 1} subtree (see annex B). The ffResponseTimeout parameter shall be available to the user of the Forward Frame service through the Cyclic Report (3.3.1.9 and 3.4.1.9) and Information Query (3.3.1.12 and 3.4.1.12) procedures.

Page 8-1

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

### 8.3 ffProdStat FUNCTIONAL RESOURCE PARAMETER

#### 8.3.1 GENERAL

The CSTS Specification Framework specifies that each CSTS provider reports the production-status using a parameter of the functional resource that represents that CSTS (see 2.2.2.2 ) of reference [1]).

**8.3.2** The production-status of the service shall be reported by the Forward Frame functional resource parameter with the classifier ffProdStat.

NOTE – The Published Identifier associated with the parameter classifier ffProdStat

is assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 1} subtree (see annex B).

**8.3.3** The ffProdStat parameter shall have the ProdStat data type specified in the SANA Functional Resource registry (reference [4])[1].

**8.3.4** The ffProdStat parameter shall be available to the user of the Forward Frame service through the Cyclic Report (3.3.1.9 and 3.4.1.9) and Information Query (3.3.1.12 and 3.4.1.12) procedures.

**8.4 ffProdStatChange EVENT**

#### 8.4.1 GENERAL

The CSTS Specification Framework specifies that each CSTS provider notifies ‘production status change’ events using an event of the functional resource that represents that CSTS, as specified in 3.11.2.2.3.2 a) of reference [1].

**8.4.2** The ‘production status change’ event of the service shall be notified with the Forward Frame functional resource event with the classifier ffProdStatChange.

NOTE – The Published Identifier associated with the event classifier ffProdStatChange is assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 2} subtree (see annex B).

**8.4.3** The eventValue component of the ffProdStatChange event shall have the

ProdStat data type specified in the SANA Functional Resource registry (reference [4])[1].

NOTE – The Published Identifier associated with the event value type for the ffProdStatChange event is assigned in the SANA Functional Resources registry (reference [4]) under the ffProdStatChange node.

**8.4.4** The ffProdStatChange event shall be available to the user of the Forward Frame service through the Notification procedure (3.3.1.15 and 3.4.1.15).

CCSDS 922.3-B-0 Page 8-2 December 2020

**8.5 ffProdConfigurationChange EVENT**

#### 8.5.1 GENERAL

The CSTS Specification Framework specifies that each CSTS provider notifies ‘production configuration change’ events using an event of the functional resource that represents that CSTS, as specified in 3.11.2.2.3.2 b) of reference [1].

**8.5.2** The ‘production configuration change’ event of the service shall be notified with the Forward Frame functional resource event with the classifier ffProdConfigurationChange.

NOTE – The Published Identifier associated with the event classifier ffProdConfigurationChange is assigned in the SANA Functional Resources registry (reference [4]) under the {ffCstsProvider 2} subtree (see annex B).

**8.5.3** The eventValue component of the ffProdConfigurationChange event shall be set to ‘empty’, as specified in 3.11.2.2.3.2 b) of reference [1].

**8.5.4** The ffProdConfigurationChange event shall be available to the user of the Forward Frame service through the Notification procedure (3.3.1.15 and 3.4.1.15).

Page 8-3

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

## 9 REFINEMENT OF DEFINITIONS OF FRAMEWORK PARAMETERS, EVENTS, DIRECTIVES, AND DIAGNOSTIC VALUES USED BY THE FORWARD FRAME SERVICE

### 9.1 GENERAL

Except where explicitly refined in this section, the definitions of the parameters, events, directives, and diagnostic values of the operations of the Framework procedures that are used by the Forward Frame service shall be the same as their definitions in reference [1].

### 9.2 ‘UNSUPPORTED OPTION’ diagnostic VALUE REFINEMENT

In addition to the definition given in 3.3.2.7.1 c) of reference [1], the ‘unsupported option’ diagnostic value shall also indicate the following error conditions:

1. a START invocation was attempted on an instance of the BFDP procedure when the Forward Frame service is configured in the ‘sequence-controlled’ data processing mode;
2. a START invocation was attempted on an instance of the Sequence-Controlled Frame Data Processing procedure when the Forward Frame service is configured in the ‘buffered’ data processing mode;
3. a START invocation was attempted on an instance of the Cyclic Report procedure when the Forward Frame service instance does not implement that procedure;
4. a START invocation was attempted on an instance of the Notification procedure when the Forward Frame service instance does not implement that procedure; and
5. an EXECUTE-DIRECTIVE invocation was attempted on an instance of the Throw Event procedure when the Forward Frame service instance does not implement that procedure.

CCSDS 922.3-B-0 Page 9-1 December 2020

**ANNEX A**

## IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA

**(NORMATIVE)**

### A1 INTRODUCTION A1.1 OVERVIEW

This annex provides the ICS Requirements List (RL) for an implementation of the *Cross Support Transfer Services—Forward Frame Service*, CCSDS 922.3-B-0, December 2020. CCSDS 922.3 specifies the requirements on the provider of the FF-CSTS.

As a member of the CSTSes suite of Recommended Standards, implementation conformance is expressed with regard to the protocol on the interface between the user and the provider of the service. Therefore the ICS is a Protocol ICS (PICS), and this annex specifies the PICS proforma RL for the Forward Frame CSTS.

The PICS for an implementation is generated by completing the RL in accordance with the instructions below. An implementation shall satisfy the mandatory conformance requirements referenced in the RL.

The RL support column in this annex is blank. An implementation’s completed RL is called the PICS. The PICS states which capabilities and options have been implemented. The following can use the PICS:

1. the implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight;
2. a supplier or potential acquirer of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
3. a user or potential user of the implementation, as a basis for initially checking the possibility of interworking with another implementation (it should be noted that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSes);
4. a tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

### A1.2 ABBREVIATIONS AND CONVENTIONS

The RL consists of information in tabular form. The status of features is indicated using the abbreviations and conventions described below.

#### Item Column

The item column contains a prefix identifying the element the given table is referring to and sequential numbers for items in the table.

#### Feature Column

The feature column contains a brief descriptive name for a feature. It implicitly means: ‘Is this feature supported by the implementation?’

#### Status Column

The status column uses the following notations:

M mandatory

O optional

O<n> optional, but support of at least one of the group of options labeled by the same numeral <n> is required

C<n> conditional as defined in corresponding expression below the table

X prohibited

N/A not applicable

#### Support Column Symbols

The support column is to be used by the implementer to state whether a feature is supported by entering Y, N, or N/A, indicating:

Y yes, supported by the implementation

N no, not supported by the implementation N/A not applicable

The support column should also be used, when appropriate, to enter values supported for a given capability.

#### Allowed Values Column

All PDU parameter types are specified in annex F of reference [1] using ASN.1. The ASN.1 data type specifications constrain among others the permissible value range, and therefore such constraints are not repeated in the Allowed Values column in the tables contained in this ICS annex. However, if a parameter is constrained for all instances of the given PDU to a subset of the range or set specified for that parameter in annex F of reference [1], then the subset is identified in the tables that contain PDU parameters.

#### Allowed Values Column Symbols

If the allowed values are too large to fit in the Allowed Values cell, the Allowed Values column uses the notation ‘AV<n>’ as an indication that the allowed values are specified below the table.

#### Supported Values Column

The Supported Values column is to be used by the implementer to state whether the specified range or set of values for the parameter is supported by entering Y or SV<n>, indicating:

Y Yes, the range/set defined in the Recommended Specification is fully supported by the implementation;

SV<n> The range/set defined in the Recommended Specification is not fully supported by the implementation. The supported subset is documented below the table.

### A1.3 INSTRUCTIONS FOR COMPLETING THE RL

An implementer shows the extent of compliance to the Recommended Standard by completing the RL; that is, the state of compliance with all mandatory requirements and the options supported are shown. The resulting completed RL is called a PICS. The implementer shall complete the RL by entering appropriate responses in the support or values supported column, using the notation described in A1.2. If a conditional requirement is inapplicable, N/A should be used. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference X*i*, where *i* is a unique identifier to an accompanying rationale for the noncompliance.

### A2 PICS PROFORMA FOR THE FORWARD FRAME CSTS PROTOCOL (CCSDS 922.1-B-1) A2.1 GENERAL INFORMATION

The PICS for a FF-CSTS implementation shall encompass the filled-in tables A-1 to A-4.

#### Table A-1: Identification of PICS

|  |  |
| --- | --- |
| Date of Statement (DD/MM/YYYY) |  |
| PICS serial number |  |
| System Conformance statement cross-reference |  |

#### Table A-2: Identification of Implementation Under Test

|  |  |
| --- | --- |
| Implementation name |  |
| Implementation version |  |
| Special Configuration |  |
| Other Information |  |

#### Table A-3: Identification of Supplier

|  |  |
| --- | --- |
| Supplier |  |
| Contact Point for Queries |  |
| Implementation Name(s) and Versions |  |
| Other information necessary for full identification, for example, name(s) and version(s) for machines and/or operating systems;    System Name(s) |  |

#### Table A-4: Identification of Specification

|  |  |  |
| --- | --- | --- |
| CCSDS 922.1-B-1 |  |  |
| Have any exceptions been required?  NOTE – A YES answer means that the implementation does not conform to the Recommended Standard. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is nonconforming. | Yes [ ] | No [ ] |

### A2.2 REQUIREMENTS LIST

This subsection provides the RLs for the elements specified in this Recommended Standard. **Table A-5: Procedures Used by the FF-CSTS Specification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Procedures** | |  |  |
| **Item** | **Description** | **Reference** | **Status** | **Support** |
| proc-1 | Association Control | 4.3 of reference [1] | M |  |
| proc-2 | Data Processing | 4.6 of reference [1] | M |  |
| proc-3 | Buffered Data Processing | 4.7 of reference [1] | C1 |  |
| proc-4 | Sequence-Controlled Data Processing | 4.8 of reference [1] | C2 |  |
| proc-5 | Information Query | 4.9 of reference [1] | O |  |
| proc-6 | Cyclic Report | 4.10 of reference [1] | O |  |
| proc-7 | Notification | 4.11 of reference [1] | O |  |
| proc-8 | Throw Event | 4.12 of reference [1] | C3 |  |
| proc-9 | Sequence-Controlled Frame Data Processing | Section 4 of this  Recommended Standard | O1 |  |
| proc-10 | Buffered Frame Data Processing | Section 5 of this  Recommended Standard | O1 |  |
| proc-11 | Master Throw Event | Section 6 of this  Recommended Standard | O |  |

C1 IF proc-10 THEN M ELSE N/A

C2 IF proc-9 THEN M ELSE N/A

C3 IF proc-11 THEN M ELSE N/A

O1 An implementation must support at least one of proc-9 or proc-10, but may support both.

NOTE – If an implementation supports both proc-9 and proc-10, one and only one procedure can be active in any given service instance, as configured by the ffDataProcessingMode service management configuration parameter specified in 3.2.

The Data Processing procedure is mandatory in the sense that the Sequence-Controlled Frame Data Processing and BFDP procedures (at least one of which is mandatory) are both derived indirectly from the Data Processing procedure. The Sequence-Controlled Data Delivery procedure is mandatory only when the Sequence-Controlled Frame Data Delivery procedure is implemented. The Buffered Data Delivery procedure is mandatory only when the Buffered Frame Data Delivery procedure is implemented. In this FF-CSTS ICS, all requirements for the Data Processing procedure are covered by the requirements for both the Sequence-Controlled Frame Data Processing and BFDP procedures, the requirements for the Sequence-Controlled Data Delivery procedure are covered by the requirements for the Sequence-Controlled Frame Data Delivery procedure, and the requirements for the Buffered Data Delivery procedure are covered by the requirements for the Buffered Frame Data Delivery procedure.

The Throw Event procedure is mandatory only when the optional Master Throw Event procedure (which is derived from the Throw Event procedure) is implemented. In this FFCSTS ICS, all requirements for the Throw Event procedure are covered by the requirements for the Master Throw Event procedure.

#### Table A-6: Required PDUs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item** | **PDU** | **Reference** | **ServiceProvider-**  **System** | | **Service-UserSystem** | |
| **Status** | **Support** | **Status** | **Support** |
| pdu-1 | BindInvocation | F3.5 of reference [1] | M |  | M |  |
| pdu-2 | BindReturn | F3.5 of reference [1] | M |  | M |  |
| pdu-3 | PeerAbortInvocation | F3.5 of reference [1] | M |  | M |  |
| pdu-4 | UnbindInvocation | F3.5 of reference [1] | M |  | M |  |
| pdu-5 | UnbindReturn | F3.5 of reference [1] | M |  | M |  |
| pdu-6 | ExecuteDirectiveAcknowledge | F3.4 of reference [1] | C4 |  | C4 |  |
| pdu-7 | ExecuteDirectiveInvocation | F3.4 of reference [1] | C4 |  | C4 |  |
| pdu-8 | ExecuteDirectiveReturn | F3.4 of reference [1] | C4 |  | C4 |  |
| pdu-9 | GetInvocation | F3.4 of reference [1] | C5 |  | C5 |  |
| pdu-10 | GetReturn | F3.4 of reference [1] | C5 |  | C5 |  |
| pdu-11 | NotifyInvocation | F3.4 of reference [1] | M |  | M |  |
| **Item** | **PDU** | **Reference** | **ServiceProvider-**  **System** | | **Service-UserSystem** | |
| **Status** | **Support** | **Status** | **Support** |
| pdu-12 | ProcessDataInvocation | F3.4 of reference [1] | M |  | M |  |
| pdu-13 | ProcessDataReturn | F3.4 of reference [1] | C6 |  | C6 |  |
| pdu-14 | StartInvocation | F3.4 of reference [1] | M |  | M |  |
| pdu-15 | StartReturn | F3.4 of reference [1] | M |  | M |  |
| pdu-16 | StopInvocation | F3.4 of reference [1] | M |  | M |  |
| pdu-17 | StopReturn | F3.4 of reference [1] | M |  | M |  |
| pdu-18 | TransferDataInvocation | F3.4 of reference [1] | C7 |  | C7 |  |
| pdu-19 | ForwardBuffer | F3.9 of reference [1] | C8 |  | C8 |  |

C4 IF proc-9 OR proc-11 THEN M ELSE N/A

C5 IF proc-5 THEN M ELSE N/A C6 IF proc-9 THEN M ELSE N/A C7 IF proc-6 THEN M ELSE N/A

C8 IF proc-10 THEN M ELSE N/A

#### Table A-7: BIND Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the BindInvocation PDU** | | |  |  | |
| **Item** | **Parameter** | **Reference** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| bindInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| bindInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| bindInv-3 | procedureName | F3.3 of reference [1] | M |  | AV1 |  |
| bindInv-4 | initiatorIdentifier | F3.5 of reference [1] | M |  |  |  |
| bindInv-5 | responderPortIdentifier | F3.5 of reference [1] | M |  |  |  |
| bindInv-6 | serviceType | F3.5 of reference [1] | M |  |  |  |
| bindInv-7 | versionNumber | F3.5 of reference [1] | M |  |  |  |
| bindInv-8 | serviceInstanceIdentifier | F3.5 of reference [1] | M |  |  |  |
| bindInv-9 | bindInvocationExtension | F3.5 of reference [1] | M |  | ‘notUsed’ |  |

AV1 For the BIND invocation the procedureRole element of the parameter bindInv-3 must be set to ‘associationControl’.

The parameters bindInv-1, bindInv-2, and bindInv-3 are contained in the complex parameter standardInvocationHeader shown in F3.5 of reference [1]. This parameter is of the type StandardInvocationHeader that is specified in F3.3 of reference [1].

#### Table A-8: BIND Return Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the BindReturn PDU** | | |  |  | |
| **Item** | **Parameter** | **Reference** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| bindRet-1 | performerCredentials | F3.3 of reference [1] | M |  |  |  |
| bindRet-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| bindRet-3 | result | F3.3 of reference [1] | M |  |  |  |
| bindRet-4 | positive | F3.3 of reference [1] | C9 |  | ‘notUsed’ |  |
| bindRet-5 | diagnostics | F3.3 of reference [1] | C10 |  | AV2 |  |
| bindRet-6 | negExtension | F3.3 of reference [1] | C10 |  | ‘notUsed’ |  |
| bindRet-7 | responderIdentifier | F3.5 of reference [1] | M |  |  |  |

C9 IF bindRet-3 = ‘positive’ THEN M ELSE X

C10 IF bindRet-3 = ‘negative’ THEN M ELSE X

AV2 For the negative BIND return, the parameter bindRet-5 is extended by the type AssocBindDiagnosticExt defined in F3.5 of reference [1]. Therefore the parameter bindRet-5 may have

1. any value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’; or
2. any value defined by ‘diagnosticExtension’: ‘acBindDiagExt’:

‘AssocBindDiagnosticExt’ defined in F3.5 of reference [1] except ‘assocBindDiagnosticExtExtension’.

All parameters of the BIND return PDU except bindRet-7 are contained in the complex parameter of the type StandardReturnHeader that is specified in F3.3 of reference [1].

Specific extensions are, however, specified in F3.5 of that document.

#### Table A-9: PEER-ABORT Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **P** | **arameters of the PeerAbortInvocation PDU** | | | |  | |
| **Item** | **Parameter** | **Reference** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| peerAbortInv-1 | diagnostic | F3.5 of reference [1] | M |  | 40..126 |  |

#### Table A-10: UNBIND Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the UnbindInvocation PDU** | | |  |  | |
| **Item** | **Parameter** | **Reference** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| unbindInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| unbindInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| unbindInv-3 | procedureName | F3.3 of reference [1] | M |  | AV3 |  |
| unbindInv-4 | unbindInvocationExtension | F3.5 of reference [1] | M |  | ‘notUsed’ |  |

AV3 For the UNBIND invocation, the procedureRole element of the parameter unbindInv-3 must be set to ‘associationControl’.

The parameters unbindInv-1, unbindInv-2, and unbindInv-3 are contained in the complex parameter standardInvocationHeader shown in F3.5 of reference [1]. This parameter is of the type StandardInvocationHeader that is specified in F3.3 of that document.

#### Table A-11: UNBIND Return Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the UnbindReturn PDU** | | |  |  | |
| **Item** | **Parameter** | **Reference** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| unbindRet-1 | performerCredentials | F3.3 of reference [1] | M |  |  |  |
| unbindRet-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| unbindRet-3 | result | F3.3 of reference [1] | M |  | AV4 |  |

AV4 The value of the parameter unbindRet-3 shall always be set to ‘positive’: ‘notUsed’; that is, the result is always positive and not extended.

All parameters of the UNBIND return PDU are contained the complex parameter of the type StandardReturnHeader that is specified in F3.3 of reference [1].

#### Table A-12: EXECUTE-DIRECTIVE Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters of the ExecuteDirectiveInvocation PDU** | | | | | | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| execDirInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| execDirInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| execDirInv-3 | procedureName | F3.3 of reference [1] | M |  | AV5 |  |
| execDirInv-4 | directiveIdentifier | F3.4 of reference [1] | M |  | AV6 |  |
| execDirInv-5 | localProcDirQualifier | F3.4 of reference [1] | C11 |  | AV7 |  |
| execDirInv-6 | targetProcedureName | F3.4 of reference [1] | X |  |  |  |
| execDirInv-7 | serviceProcDirQualifierValues | F3.4 of reference [1] | X |  |  |  |
| execDirInv-8 | functionalResourceInstanceNu mber | F3.4 of reference [1] | C12 |  |  |  |
| execDirInv-9 | functionalResourceQualifiers | F3.4 of reference [1] | C12 |  | AV8 |  |
| execDirInv-10 | directiveQualifierExtension | F3.4 of reference [1] | X |  |  |  |
| execDirInv-11 | executeDirectiveInvocationExte nsion | F3.4 of reference [1] | M |  | ‘notUsed’ |  |

C11 IF execDirInv-4 is set to the Published Identifier of the ‘reset’ directive (pSCDPresetDirective as defined in F3.16 of reference [1]) THEN M ELSE X

C12 IF execDirInv-4 is set to the Published Identifier of a directive that is registered under a FR type THEN M ELSE X

AV5 If the procedureType element of the parameter execDirInv-3 has the value ‘sequenceControlledFrameDataProcessing’, the value of the procedureRole element of the parameter startInv-3 must be set to ‘prime procedure’; otherwise the value of the procedureRole element of the parameter execDirInv-3 must be set to ‘secondary procedure’.

AV6 The Published Identifier specified in the execDirInv-4 parameter must identify a registered directive.

AV7 The parameter execDirInv-5 will be: ‘directiveQualifier’: ‘localProcDirQualifier’:

‘DirectiveQualifierValues’: ‘parameterlessValues’: ‘TypeAndValue’: Embedded’: ‘EMBEDDED PDV’, where the Object Identifier and the type of the directivequalifier parameter are pSCDPresetDirectiveDirQual and

PSCDPresetDirectiveDirQual, as specified in F3.16 of reference [1].

AV8 The parameter execDirInv-9 will be one of the following:

1. ‘directiveQualifier’: ‘functResourceDirQualifier’:

‘functionalResourceQualifiers’: ‘DirectiveQualifierValues’: ‘sequenceOfParamIdsAndValues’,

1. ‘directiveQualifier’: ‘functResourceDirQualifier’:

‘functionalResourceQualifiers’: ‘DirectiveQualifierValues’: ‘parameterlessValues’, or

1. ‘directiveQualifier’: ‘functResourceDirQualifier’:

‘functionalResourceQualifiers’: ‘DirectiveQualifierValues’: ‘noQualifierValues’.

The parameters execDirInv-1, execDirInv-2, and execDir-3 are contained in the complex parameter standardInvocationHeader in the ExecuteDirectiveInvocation type shown in F3.4 of reference [1]. This parameter is of the type

StandardInvocationHeader that is specified in F3.3 of reference [1].

#### Table A-13: EXECUTE-DIRECTIVE Acknowledgement Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the ExecuteDirectiveAcknowledge PDU** | | | | | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| execDirAck-1 | performerCredentials | F3.3 of reference [1] | M |  |  |  |
| execDirAck-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| execDirAck-3 | result | F3.3 of reference [1] | M |  |  |  |
| execDirAck-4 | positive | F3.3 of reference [1] | C13 |  | ‘notUsed’ |  |
| execDirAck-5 | diagnostic | F3.3 of reference [1] | C14 |  | AV9 |  |
| execDirAck-6 | negExtension | F3.3 of reference [1] | C14 |  | ‘notUsed’ |  |

C13 IF execDirAck-3 = ‘positive’ THEN M ELSE X

C14 IF execDirAck-3 = ‘negative’ THEN M ELSE X

AV9 For the EXECUTE-DIRECTIVE acknowledgement the parameter execDirAck-5 is extended by the type ExecDirNegAckDiagnosticExt defined in F3.4 of reference [1]. Therefore the parameter execDirAck-5 may have

1. any value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’; or
2. any value defined by the extension ‘diagnosticExtension’:

‘execDirAckDiagExt’: ‘ExecDirNegAckDiagnosticExt’ in F3.4 of reference [1] except ‘execDirNegAckDiagnosticExtExtension’.

All parameters of the EXECUTE-DIRECTIVE acknowledgement PDU are contained the complex parameter of the type StandardReturnHeader that is specified in F3.3 of reference [1]. Specific extensions are, however, specified in F3.4 of reference [1].

**Table A-14: EXECUTE-DIRECTIVE Return Parameters**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the ExecuteDirectiveReturn PDU** | | | | | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| execDirRet-1 | performerCredentials | F3.3 of reference [1] | M |  |  |  |
| execDirRet-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| execDirRet-3 | result | F3.3 of reference [1] | M |  |  |  |
| execDirRet-4 | positive | F3.3 of reference [1] | C15 |  | ‘notUsed’ |  |
| execDirRet-5 | diagnostic | F3.3 of reference [1] | C16 |  | AV10 |  |
| execDirRet-6 | negExtension | F3.3 of reference [1] | C16 |  | ‘notUsed’ |  |

C15 IF execDirRet-3 = ‘positive’ THEN M ELSE X

C16 IF execDirRet-3 = ‘negative’ THEN M ELSE X

AV10 For the negative EXECUTE-DIRECTIVE return PDU the parameter execDirRet5 is extended by the type ExecDirNegReturnDiagnosticExt defined in F3.4 of reference [1]. Therefore the parameter execDirRet-5 may have

1. any standard value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’; or
2. any value defined by the extension ‘diagnosticExtension’:

‘execDirNegReturnDiagnosticExt’: ‘ExecDirNegReturnDiagnosticExt’ defined in F3.4 of reference [1] except ‘execDirNegReturnDiagnosticExtExtension’. Additional values can be introduced by the further extension ‘diagnosticExtension’: ‘execDirNegReturnDiagnosticExt’:

‘ExecDirNegReturnDiagnosticExt’:

‘execDirNegReturnDiagnosticExtExtension’.

If the EXECUTE-DIRECTIVE return PDU is used by the Master Throw Event procedure, that is, the procedureType element of the parameter execDirInv-3 of the associated EXECUTE-DIRECTIVE invocation has the value

‘masterThrowEvent’, additional values are introduced by the further extension ‘diagnosticExtension’: ‘execDirNegReturnDiagnosticExt’:

‘ExecDirNegReturnDiagnosticExt’:

‘execDirNegReturnDiagnosticExtExtension’: ‘teExecDirDiagExt’:

‘TeExecDirNegReturnDiagnosticExt’ (the type

TeExecDirNegReturnDiagnosticExt is specified in F3.14 of reference [1]). Therefore the parameter execDirRet-5 may have, in this case,

1. any standard value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’;
2. any value defined by the extension ‘diagnosticExtension’:

‘execDirNegReturnDiagnosticExt’: ‘ExecDirNegReturnDiagnosticExt’ defined in F3.4 of reference [1] except ‘execDirNegReturnDiagnosticExtExtension’; or

1. any value defined by the extension ‘diagnosticExtension’:

‘execDirNegReturnDiagnosticExt’: ‘ExecDirNegReturnDiagnosticExt’:

‘execDirNegReturnDiagnosticExtExtension’: ‘teExecDirDiagExt’:

‘TeExecDirNegReturnDiagnosticExt’ (defined in F3.14 of reference [1]) except ‘teExecDirNegReturnDiagnosticExtExtension’.

All parameters of the EXECUTE-DIRECTIVE return PDU are contained the complex parameter of the type StandardReturnHeader that is specified in F3.3 of reference [1].

Specific extensions are, however, specified in F3.4 and F3.14 of reference [1].

#### Table A-15: GET Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the GetInvocation PDU** | | |  |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| getInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| getInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| getInv-3 | procedureName | F3.3 of reference [1] | M |  | AV11 |  |
| getInv-4 | listOfParameters | F3.4 of reference [1] | M |  |  |  |
| getInv-5 | getInvocationExtension | F3.4 of reference [1] | M |  | ‘notUsed’ |  |

AV11 The value of the procedureRole element of the parameter getInv-3 is constrained to have the value, ‘secondary procedure’.

The parameters getInv-1, getInv-2, and getInv-3 are contained in the complex parameter standardInvocationHeader in the GetInvocation type shown in F3.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is specified in F3.3 of reference [1].

#### Table A-16: GET Return Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the GetReturn PDU** | | |  |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| getRet-1 | performerCredentials | F3.3 of reference [1] | M |  |  |  |
| getRet-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| getRet-3 | result | F3.3 of reference [1] | M |  |  |  |
| getRet-4 | positive | F3.3 of reference [1] | C17 |  | AV12 |  |
| getRet-5 | qualifiedParameters | F3.4 of reference [1] | C17 |  | AV13 |  |
| getRet-6 | getPosReturnExtExtensi on | F3.4 of reference [1] | C17 |  | ‘notUsed’ |  |
| getRet-7 | diagnostic | F3.3 of reference [1] | C18 |  | AV14 |  |
| getRet-8 | negExtension | F3.3 of reference [1] | C18 |  | ‘notUsed’ |  |

C17 IF getRet-3 = ‘positive’ THEN M ELSE X

C18 IF getRet-3 = ‘negative’ THEN M ELSE X

AV12 For the positive GET return, the parameter getRet-4 is set to ‘getPosReturnExt’: ‘GetPosReturnExt’ defined in F3.4 of reference [1].

AV13 For the positive GET return, the parameter getRet-5 is specified by ‘qualifiedParameters’: ‘QualifiedParametersSequence’. The type

QualifiedParametersSequence is defined in F3.4 of reference [1].

AV14 For the negative GET return, the parameter getRet-7 is extended by the type GetDiagnosticExt defined in F3.4 of reference [1]. Therefore the parameter getRet-7 may have

1. any standard value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’; or
2. any value defined by the extension ‘diagnosticExtension’:

‘getDiagnosticExt’: ‘GetDiagnosticExt’ defined in F3.4 of reference [1] except ‘getDiagnosticExtExtension’.

All parameters of the GET return PDU are contained the complex parameter of the type StandardReturnHeader that is specified in F3.3 of reference [1]. Specific extensions are, however, specified in F3.4 of reference [1].

#### Table A-17: PROCESS-DATA Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters of the ProcessDataInvocation PDU** | | | | | | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| procDataInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| procDataInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| procDataInv-3 | procedureName | F3.3 of reference [1] | M |  | AV15 |  |
| procDataInv-4 | dataUnitId | F3.4 of reference [1] | M |  |  |  |
| procDataInv-5 | data | F3.4 of reference [1] | M |  | AV16 |  |
| procDataInv-6 | processDataInvoca tionExtension | F3.8 of reference [1] | M |  | AV17 |  |
| procDataInv-7 | processCompletion Report | F3.8 of reference [1] | M |  |  |  |
| procDataInv-8 | dataProcProcDataI nvocExtExtension | F3.8 of reference [1] | M |  | AV18 |  |
| procDataInv-9 | earliestDataProces singTime | F3.10 of reference [1] | C19 |  |  |  |
| procDataInv-10 | latestDataProcessi ngTime | F3.10 of reference [1] | C19 |  |  |  |
| procDataInv-11 | sequContrDataPro cProcDataInvocExt Extension | F3.10 of reference [1] | C19 |  | ‘notUsed’ |  |

C19 IF procDataInv-8 = (‘scdpProcDataInvocExt’:

‘SequContrDataProcProcDataInvocExt’) THEN M ELSE X

AV15 The value of the procedureRole element of the parameter startInv-3 must be set to ‘prime procedure’.

AV16 The parameter procDataInv-5 is refined to be an octet string.

AV17 The parameter procDataInv-6 shall be set to the value ‘dpProcDataInvocExt’:

‘DataProcProcDataInvocExt’.

AV18 If the PDU is used by the BFDP procedure, that is, the procedureType element of the parameter procDataInv-3 has the value ‘BufferedFrameDataProcessing’, the value of this parameter shall be set to ‘notUsed’.

If the PDU is used by the Sequence-Controlled Frame Data Processing procedure, that is, the procedureType element of the parameter procDataInv-3 has the value ‘sequenceControlledFrameDataProcessing’, then the parameter procDataInv-8 shall be set to ‘scdpProcDataInvocExt’: ‘SequContrDataProcProcDataInvocExt’.

The parameters procDataInv-1, procDataInv-2, and procDataInv-3 are contained in the complex parameter standardInvocationHeader in the ProcessDataInvocation type shown in F3.4 of reference [1]. This parameter is of the type

StandardInvocationHeader that is in subsection F3.3 of reference [1].

#### Table A-18: PROCESS-DATA Return Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the ProcessDataReturn PDU** | | | |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| procDataRet-1 | performerCredential s | F3.3 of reference [1] | M |  |  |  |
| procDataRet-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| procDataRet-3 | result | F3.3 of reference [1] | M |  |  |  |
| procDataRet-4 | positive | F3.3 of reference [1] | C20 |  | AV19 |  |
| procDataRet-5 | dataUnitId | F3.10 of reference [1] | C20 |  |  |  |
| procDataRet-6 | sequContrDataProc ProcDataPosReturn  ExtExtension | F3.10 of reference [1] | C20 |  | ‘notUsed’ |  |
| procDataRet-7 | diagnostic | F3.3 of reference [1]  and annex C of  this  Recommended  Standard | C21 |  | AV20 |  |
| procDataRet-8 | negExtension | F3.3 of reference [1] | C21 |  | AV21 |  |
| procDataRet-9 | dataUnitId | F3.10 of reference [1] | C21 |  |  |  |
| procDataRet-10 | sequContrDataProc ProcDataNegReturn  ExtExtension | F3.10 of reference [1] | C21 |  | ‘notUsed’ |  |

C20 IF procDataRet-3 = ‘positive’ THEN M ELSE X

C21 IF procDataRet-3 = ‘negative’ THEN M ELSE X

AV19 For the Forward Frame service, this PDU is used only by the Sequence-Controlled Frame Data Processing procedure, that is, the procedureType element of parameter procDataInv-3 in the associated invocation has the value ‘sequenceControlledFrameDataProcessing’. Therefore the parameter procDataRet-4 shall be set to ‘scdpProcDataPosReturnExt’:

‘SequContrDataProcProcDataPosReturnExt’, defined in F3.10 of reference [1].

AV20 For the PROCESS-DATA return, the parameter procDataRet-7 is extended by the type SequContrDataProcProcDataDiagnosticExt defined in F3.10 of reference [1]. Therefore the parameter procDataRet-7 may have:

1. any standard value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’; or
2. any value defined by the extension ‘diagnosticExtension’:

‘scdpProcDataDiagExt’: ‘SequContrDataProcProcDataDiagnosticExt’ defined in

F3.10 of reference [1] except

‘sequContrDataProcProcDataDiagnosticExtExtension’; or

‘SequContrFrameDataProcProcDataDiagnosticExt’ defined in annex C except ‘scfdpProcDataDiagnosticExtExtension’.

AV21 For the Forward Frame service, this PDU is used only by the Sequence-Controlled Frame Data Processing procedure; that is, the procedureType element of parameter procDataInv-3 in the associated invocation has the value ‘sequenceControlledFrameDataProcessing’. Therefore the parameter procDataRet-8 shall be set to ‘scdpProcDataNegReturnExt’:

‘SequContrDataProcProcDataNegReturnExt’. The type

SequContrDataProcProcDataNegReturnExt is defined in F3.10 of reference [1].

This PDU is valid only in the case that the PROCESS-DATA operation is used by the Sequence-Controlled Frame Data Processing procedure; that is, the procedureType element of the parameter procDataInv-3 of the associated PROCESS-DATA invocation has the value ‘sequenceControlledFrameDataProcessing’. The BFDP procedure uses the unconfirmed variant of the PROCESS-DATA operation.

All parameters of the PROCESS-DATA return PDU are contained the complex parameter of the type StandardReturnHeader that is specified in F3.3 of reference [1]. Specific extensions are, however, specified in F3.10 of reference [1] and annex C.

#### Table A-19: START Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the StartInvocation PDU** | | | |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| startInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| startInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| startInv-3 | procedureName | F3.3 of reference [1] | M |  | AV22 |  |
| startInv-4 | startInvocationExtension | F3.4 of reference [1] | M |  | AV23 |  |
| startInv-5 | firstDataUnitId | F3.10 of reference [1] | C22 |  |  |  |
| startInv-6 | sequContrDataProcStartInvoc ExtExtension | F3.10 of reference [1] | C22 |  | AV24 |  |
| startInv-7 | sequContrFrameInputQueueS ize | annex C | C22 |  |  |  |
| startInv-8 | sequContrFrameDataProcSta rtInvocExtExtension | annex C | C22 |  | ‘notUsed’ |  |
| startInv-9 | buffFrameInputQueueSize | annex D | C23 |  |  |  |
| startInv-10 | buffFrameMaximumForwardB  ufferSize | annex D | C23 |  |  |  |
| startInv-11 | buffFrameprocessingLatencyL imit | annex D | C23 |  |  |  |
| startInv-12 | buffFrameDataProcStartInvoc ExtExtension | annex D | C23 |  | ‘notUsed’ |  |
| startInv-13 | deliveryCycle | F3.12 of reference [1] | C24 |  |  |  |
| startInv-14 | listOfParameters | F3.12 of reference [1] | C24 |  |  |  |
| startInv-15 | cyclicReportStartInvocExtExte nsion | F3.12 of reference [1] | C24 |  | ‘notUsed’ |  |
| startInv-16 | listOfEvents | F3.13 of reference [1] | C25 |  |  |  |
| startInv-17 | notificationStartInvocExtExten sion | F3.3 of reference [1] | C25 |  | ‘notUsed’ |  |

C22 IF startInv-4 = ‘scdpStartInvocExt’: ‘SequContrDataProcStartInvocExt’ THEN

M ELSE X

C23 IF startInv-4 = ‘buffFrameDataProcStartInvocExt’:

‘BuffFrameDataProcStartInvocExt’ THEN M ELSE X

C24 IF startInv-4 = ‘crStartInvocExt’: ‘CyclicReportStartInvocExt’ THEN M ELSE X

C25 IF startInv-4 = ‘nStartInvocExt’: ‘NotificationStartInvocExt’ THEN M ELSE X

AV22 If the procedureType element of the parameter startInv-3 has the value ‘sequenceControlledFrameDataProcessing’ or ‘bufferedFrameDataProcessing’, the value of the procedureRole element of the parameter startInv-3 must be set to ‘prime procedure’; otherwise, the value of the procedureRole element of the parameter startInv-3 must be set to ‘secondary procedure’.

AV23 If the procedureType element of the parameter startInv-3 has the value

‘sequenceControlledFrameDataProcessing’, then the parameter startInv-4 shall be set to the value ‘scdpStartInvocExt’: ‘SequContrDataProcStartInvocExt’. If the procedureType element of the parameter startInv-3 has the value ‘bufferedFrameDataProcessing’, then the parameter startInv-4 shall be set to the value ‘buffFrameDataProcStartInvocExt’: ‘BuffFrameDataProcStartInvocExt’. If the procedureType element of the parameter startInv-3 has the value ‘cyclicReport’, then the parameter startInv-4 shall be set to the value ‘crStartInvocExt’: ‘CyclicReportStartInvocExt’.

If the procedureType element of the parameter startInv-3 has the value ‘notification’, then the parameter startInv-4 shall be set to the value ‘nStartInvocExt’: ‘NotificationStartInvocExt’.

AV24 The parameter startInv-6 shall be set to the value ‘sequContrFrameDataProcStartInvocExt’: SequContrFrameDataProcStartInvocExt’.

The parameters startInv-1, startInv-2, and startInv-3 are contained in the complex parameter standardInvocationHeader in the StartInvocation type shown in F3.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is specified in F3.3 of reference [1].

**Table A-20: START Return Parameters**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the StartReturn PDU** | | | |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| startRet-1 | performerCredentials | F3.3 of reference [1] | M |  |  |  |
| startRet-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| startRet-3 | result | F3.3 of reference [1] | M |  |  |  |
| startRet-4 | positive | F3.3 of reference [1] | C26 |  | ‘notUsed’ |  |
| startRet-5 | diagnostic | F3.3 of reference [1] | C27 |  | AV25 |  |
| startRet-6 | negExtension | F3.3 of reference [1] | C27 |  | ‘notUsed’ |  |

C26 IF startRet-3 = ‘positive’ THEN M ELSE X

C27 IF startRet-3 = ‘negative’ THEN M ELSE X

AV25 If the procedureType element of the parameter startInv-3 of the associated

START invocation has the value of either

‘sequenceControlledFrameDataProcessing’ or ‘BufferedFrameDataProcessing’, then parameter startRet-5 is extended only by the type StartDiagnosticExt defined in F3.4 of reference [1]. Therefore the parameter startRet-5 may have

1. any standard value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’; or
2. any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’: ‘StartDiagnosticExt’ in F3.3 of reference [1] except ‘startDiagnosticExtExtension’.

If the procedureType element of the parameter startInv-3 of the associated START invocation has the value ‘cyclicReport’, then parameter startRet-5 is extended by the type CyclicReportStartDiagnosticExt defined in F3.12 of reference [1]. Therefore the parameter startRet-5 may have in this case

(a) any standard value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’;

b) any value defined by the extension ‘diagnosticExtension’:

‘startDiagnosticExt’: ‘StartDiagnosticExt’ in subsection F3.4 of reference [1] except ‘startDiagnosticExtExtension’; or

(c) any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’: ‘StartDiagnosticExt’: ‘startDiagnosticExtExtension’: crStartDiagExt’: ‘CyclicReportStartDiagnosticExt’ defined in F3.12 of reference [1] except ‘cyclicReportStartDiagnosticExtExtension’.

If the procedureType element of the parameter startInv-3 of the associated START invocation has the value ‘notification’, then parameter startRet-5 is extended by the type NotificationStartDiagnosticExt defined in F3.13 of reference [1]. Therefore the parameter startRet-5 may have in this case

1. any standard value defined for the Diagnostic type in subsection F3.3 of reference [1] except ‘diagnosticExtension’;
2. any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’: ‘StartDiagnosticExt’ in F3.4 of reference [1] except ‘startDiagnosticExtExtension’; or
3. any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’: ‘StartDiagnosticExt’: ‘startDiagnosticExtExtension’: ‘nStartDiagExt’: ‘NotificationStartDiagnosticExt’ defined in F3.3 of reference [1] except ‘notificationStartDiagnosticExtExtension’.

#### Table A-21: STOP Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the StopInvocation PDU** | | | |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| stopInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| stopInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| stopInv-3 | procedureName | F3.3 of reference [1] | M |  | AV26 |  |
| stopInv-4 | stopInvocationExtension | F3.4 of reference [1] | M |  | ‘notUsed’ |  |

AV26 If the procedureType element of the parameter stopInv-3 has the value ‘sequenceControlledFrameDataProcessing’ or ‘bufferedFrameDataProcessing’, the value of the procedureRole element of the parameter stopInv-3 must be set to ‘prime procedure’; otherwise, the value of the procedureRole element of the parameter startInv-3 must be set to ‘secondary procedure’.

The parameters stopInv-1, stopInv-2, and stopInv-3 are contained in the complex parameter standardInvocationHeader in the StopInvocation type shown in F3.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is specified in F3.3 of reference [1].

#### Table A-22: STOP Return Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the StopReturn PDU** | | |  |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| stopRet-1 | performerCredentials | F3.3 of reference [1] | M |  |  |  |
| stopRet-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| stopRet-3 | result | F3.3 of reference [1] | M |  |  |  |
| stopRet-4 | positive | F3.3 of reference [1] | C28 |  | ‘notUsed’ |  |
| stopRet-5 | diagnostic | F3.3 of reference [1] | C29 |  | AV27 |  |
| stopRet-6 | negExtension | F3.3 of reference [1] | C29 |  | ‘notUsed’ |  |

C28 IF stopRet-3 = ‘positive’ THEN M ELSE X

C29 IF stopRet-3 = ‘negative’ THEN M ELSE X

AV27 The parameter stopRet-5 may have any standard value defined for the Diagnostic type in F3.3 of reference [1] except ‘diagnosticExtension’.

All parameters of the STOP return PDU are contained the complex parameter of the type StandardReturnHeader that is specified in F3.3 of reference [1].

#### Table A-23: NOTIFY Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Parameters of the NotifyInvocation PDU** | | | |  | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| notifyInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| notifyInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| notifyInv-3 | procedureName | F3.3 of reference [1] | M |  | AV28 |  |
| notifyInv-4 | eventTime | F3.4 of reference [1] | M |  |  |  |
| notifyInv-5 | eventName | F3.4 of reference [1] | M |  |  |  |
| notifyInv-6 | eventValue | F3.4 of reference [1] | M |  | AV29 |  |
| notifyInv-7 | notifyInvocationExtensi on | F3.4 of reference [1] | M |  | AV30 |  |
| notifyInv-8 | dataUnitIdLastProcesse d | F3.8 of reference [1] | C30 |  |  |  |
| notifyInv-9 | dataProcessingStatus | F3.8 of reference [1] | C31 |  | AV31 |  |
| notifyInv-10 | dataProcessingStartTi me | F3.8 of reference [1] | C31 |  |  |  |
| notifyInv-11 | dataUnitIdLastOk | F3.8 of reference [1] | C30 |  |  |  |
| notifyInv-12 | dataProcessingStopTim e | F3.8 of reference [1] | C32 |  |  |  |
| notifyInv-13 | productionStatus | F3.8 of reference [1] | C30 |  | AV32 |  |
| notifyInv-14 | dataProcNotifyInvocExt Extension | F3.8 of reference [1] | C30 |  | ‘notUsed’ |  |

C30 IF notifyInv-7 = ‘dpNotifyInvocExt’: ‘DataProcNotifyInvocExt’ THEN M ELSE X

C31 IF (notifyInv-7 = ‘dpNotifyInvocExt’: ‘DataProcNotifyInvocExt’) AND (notifyInv8 is not ‘noDataProcessed’) THEN M ELSE X

C32 IF (notifyInv-7 = ‘dpNotifyInvocExt’: ‘DataProcNotifyInvocExt’) AND (notifyInv11 is not ‘noSuccessfulProcessing’) THEN M ELSE X

AV28 If the procedureType element of the parameter notifyInv-3 has the value ‘sequenceControlledFrameDataProcessing’ or ‘bufferedFrameDataProcessing’, the value of the procedureRole element of the parameter stopInv-3 must be set to ‘prime procedure’; otherwise, the value of the procedureRole element of the parameter notifyInv-3 must be set to ‘secondary procedure’.

AV29 If the procedureType element of the parameter notifyInv-3 has the value ‘notification’, the value of the notifyInv-6 parameter can be a value that can be expressed using the type SequenceOfQualifiedValue specified in F3.3 of reference [1] or ‘empty’, but must not be set to ‘eventValueExtension’ (see EventValue defined in F3.3 of reference [1]).

If the procedureType element of the parameter notifyInv-3 has the value ‘sequenceControlledFrameDataProcessing’, the value of the notifyInv-6 parameter can be any of the event values that are specified for the events specified in (4.6.4.2.3, 4.8.4.3.3 or 4.8.4.3.4 of reference [1]).

If the procedureType element of the parameter notifyInv-3 has the value ‘bufferedFrameDataProcessing’, the value of the notifyInv-6 parameter can be any of the event values that are specified for the events defined in (4.6.4.2.3 or 4.7.4.2.2 of reference [1]) or 5.6.7.2.1 of this Recommended Standard.

AV30 If the procedureType element of the parameter notifyInv-3 has the value ‘bufferedFrameDataProcessing’ or ‘sequenceControlledFrameDataProcessing’, then the parameter notifyInv-7 shall be set to the value ‘dpNotifyInvocExt’: ‘DataProcNotifyInvocExt’.

In all other cases, this parameter shall be set to ‘notUsed’.

AV31 If the procedureType element of the parameter notifyInv-3 has the value

‘sequenceControlledFrameDataProcessing’, the parameter notifyInv-9 can take on

1. any value specified for the dataProcessingStatus element of the DataProcNotifyInvocExt type defined in F3.8 of reference [1] except dataProcessingStatusExtension; or
2. any value specified by the SequContrDataProcStatus type defined in F3.8 of reference [1] except sequContrDataProcStatusExtension.

If the procedureType element of the parameter notifyInv-3 has the value ‘bufferedFrameDataProcessing’, the parameter notifyInv-9 can take on any value specified for the dataProcessingStatus element of the DataProcNotifyInvocExt type defined in F3.8 of reference [1] except dataProcessingStatusExtension.

AV32 If the parameter notifyInv-5 has the value ‘productionStatusChange’, the parameter notifyInv-13 shall be set to ‘productionStatusChange’: ‘NULL’; that is, it is absent. For all other values of the parameter notifyInv-5, it shall be set to ‘anyOtherEvent’: ‘ProductionStatus’.

The parameters notifyInv-1, notifyInv-2, and notifyInv-3 are contained in the complex parameter standardInvocationHeader in the NotifyInvocation type shown in F3.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is specified in F3.3 of reference [1].

#### Table A-24: TRANSFER-DATA Invocation Parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Parameters of the TransferDataInvocation PDU** | | | | | | |
| **Item** | **Parameter** | **Ref.** | **Status** | **Support** | **Values** | |
| **Allowed** | **Supported** |
| transferDataInv-1 | invokerCredentials | F3.3 of reference [1] | M |  |  |  |
| transferDataInv-2 | invokeId | F3.3 of reference [1] | M |  |  |  |
| transferDataInv-3 | procedureName | F3.3 of reference [1] | M |  | AV33 |  |
| transferDataInv-4 | generationTime | F3.4 of reference [1] | M |  |  |  |
| transferDataInv-5 | sequenceCounter | F3.4 of reference [1] | M |  |  |  |
| transferDataInv-6 | data | F3.4 of reference [1] | M |  | AV34 |  |
| transferDataInv-7 | qualifiedParameters | F3.12 of reference [1] | M |  |  |  |
| transferDataInv-8 | cyclicReportTransferDat aInvocDataRefExtensio  n | F3.12 of reference [1] | M |  | ‘notUsed’ |  |
| transferDataInv-9 | transferDataInvocationE xtension | F3.4 of reference [1] | M |  | ‘notUsed’ |  |

AV33 The value of the procedureRole element of the parameter transferDataInv-3 shall be set to ‘secondary procedure’.

AV34 The parameter transferDataInv-6 shall be set to the value ‘extendedData’: ‘crTransferDataInvocDataRef’: ‘CyclicReportTransferDataInvocDataRef’. The type CyclicReportTransferDataInvocDataRef is defined in F3.12 of reference [1].

The parameters transferDataInv-1, transferDataInv-2, and transferDataInv-3 are contained in the complex parameter standardInvocationHeader in the TransferDataInvocation type shown in F3.4 of reference [1]. This parameter is of the type

StandardInvocationHeader that is specified in subsection F3.3 of reference [1].

**ANNEX B**

**SERVICE OBJECT IDENTIFIERS**

**(NORMATIVE)**

### B1 FORWARD FRAME OBJECT IDENTIFIERS ASN.1 MODULE

CCSDS-FORWARD-FRAME-OBJECT-IDENTIFIERS

{ iso(1) identified-organization(3) standards-producing-organization(112) ccsds(4) css(4) csts(1) services(2) forwardFrame(3)

forwardFrameServiceModules(4) object-identifiers(1) version (1)

}

DEFINITIONS

IMPLICIT TAGS

::= BEGIN

EXPORTS bfdpDirectivesId

, bfdpEventsId

, bfdpExtendedOpsParam

, bfdpExtendedProcParam

, forwardFrameDerivedServices

, forwardFrameExtendedServiceParameters

, forwardFrameServiceProcedures

, masterThrowEventDirectivesId

, masterThrowEventEventsId

, masterThrowEventExtendedOpsParam

, masterThrowEventExtendedProcParam

, scfdpExtendedOpsParam

, scfdpExtendedProcParam

, scfdpEventsId

, scfdpDirectivesId

**;**

IMPORTS crossSupportFunctionalities

, services

FROM CCSDS-CSTS-OBJECT-IDENTIFIERS

BufferSize

, IntUnsigned

FROM CCSDS-CSTS-COMMON-TYPES

;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Root Object Identifiers of the Service

forwardFrame OBJECT IDENTIFIER ::= {services 3}

forwardFrameDerivedServices OBJECT IDENTIFIER ::= {forwardFrame 1}

forwardFrameExtendedServiceParameters OBJECT IDENTIFIER ::=

{forwardFrame 2}

forwardFrameServiceProcedures OBJECT IDENTIFIER ::=

{forwardFrame 3}

forwardFrameServiceModules OBJECT IDENTIFIER ::= {forwardFrame 4}

Page B-1

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Procedure Type Identifiers

sequenceControlledFrameDataProcessing OBJECT IDENTIFIER

::= {forwardFrameServiceProcedures 1} scfdpExtendedOpsParam OBJECT IDENTIFIER

::= {sequenceControlledFrameDataProcessing 1} scfdpExtendedProcParam OBJECT IDENTIFIER

::= {sequenceControlledFrameDataProcessing 2} scfdpEventsId OBJECT IDENTIFIER

::= {sequenceControlledFrameDataProcessing 3} scfdpDirectivesId OBJECT IDENTIFIER

::= {sequenceControlledFrameDataProcessing 4}

bufferedFrameDataProcessing OBJECT IDENTIFIER

::= {forwardFrameServiceProcedures 2}

bfdpExtendedOpsParam OBJECT IDENTIFIER ::= {bufferedFrameDataProcessing 1} bfdpExtendedProcParam OBJECT IDENTIFIER ::= {bufferedFrameDataProcessing 2} bfdpEventsId OBJECT IDENTIFIER ::= {bufferedFrameDataProcessing 3} bfdpDirectivesId OBJECT IDENTIFIER ::= {bufferedFrameDataProcessing 4}

masterThrowEvent OBJECT IDENTIFIER

::= {forwardFrameServiceProcedures 3}

masterThrowEventExtendedOpsParam OBJECT IDENTIFIER

::= {masterThrowEvent 1}

masterThrowEventExtendedProcParam OBJECT IDENTIFIER

::= {masterThrowEvent 2} masterThrowEventEventsId OBJECT IDENTIFIER ::= {masterThrowEvent 3}

masterThrowEventDirectivesId OBJECT IDENTIFIER

::= {masterThrowEvent 4}

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Object Identifiers and Data Types of the FF-CSTS Provider

-- FR type

-- The Object Identifiers of the FF-CSTS Provider FR type

-- are specified and registered in the SANA Functional Resources registry

-- (reference [4]) under the crossSupportFunctionalities subtree. -- The root of the FF-CSTS Provider FR type OID has the

-- classifier ffCstsProvider.

-- All parameters of the FF-CSTS Provider FR shall be

-- registered under the node of the ffCstsProvider subtree that has the

-- Object Identifier {ffCstsProvider 1}.These include,but are not limited

-- to, the service management parameters and read-only functional resource

-- parameters specified in 4.8 and 5.8:

-- All events of the FF-CSTS Provider FR shall be

-- registered under the node of the ffCstsProvider subtree that has the

-- Object Identifier {ffCstsProvider 2}.

-- All directives of the FF-CSTS Provider FR shall be

-- registered under the node of the ffCstsProvider subtree that has the -- Object Identifier {ffCstsProvider 3}.

END

Page B-2

### B2 TRANSFER SYNTAX

There are no data types specified in this module.

Page B-3

**ANNEX C**

**PROCEDURE—SEQUENCE-CONTROLLED FRAME DATA**

**PROCESSING PDUS**

**(NORMATIVE)**

### C1 SEQUENCE-CONTROLLED FRAME DATA PROCESSING PDUS ASN.1 MODULE

CCSDS-SEQUENCE-CONTROLLED-FRAME-DATA-PROCESSING-PDUS

{ iso(1) identified-organization(3) standards-producing-organization(112) ccsds(4) css(4) csts(1) services(2) forwardFrame(3) forwardFrameServiceModules(4) extensions(2)

sequenceControlledFrameDataProcessingPdus(1) version (1)

}

DEFINITIONS

IMPLICIT TAGS

::= BEGIN

EXPORTS SequContrFrameDataProcProcDataDiagnosticExt

, SequContrFrameDataProcStartInvocExt

, SequenceControlledFrameDataProcessingPdu

;

IMPORTS AdditionalText

, Embedded

, Extended

, IntUnsigned

FROM CCSDS-CSTS-COMMON-TYPES

-- CCSDS-CSTS-COMMON-TYPES is defined in F3.3 of the -- CSTS SFW (reference [1]).

SequContrDataProcessingPdu

FROM CCSDS-CSTS-SEQUENCE-CONTROLLED-DATA-PROCESSING-PDUS

-- CCSDS-CSTS-SEQUENCE-CONTROLLED-DATA-PROCESSING-PDUS is defined in F3.10 -- of the CSTS SFW (reference [1]).

scfdpExtendedOpsParam

FROM CCSDS-FORWARD-FRAME-OBJECT-IDENTIFIERS

;

-- The Sequence-Controlled Frame Data Processing procedure is derived from

-- the CSTS SFW Sequence-Controlled Data Processing procedure, which

-- in turn is derived from the Data Processing procedure. Its PDU

-- is cast as the type of the PDU defined in the Sequence-Controlled Frame

Data Processing procedure:

-- SequContrDataProcessingPdu type defined in the

-- CCSDS-CSTS-SEQUENCE-CONTROLLED-DATA-PROCESSING-PDUS module -- (F3.10) of the CSTS SFW (reference [1]).

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

SequenceControlledFrameDataProcessingPdu ::= SequContrDataProcessingPdu

-- \*\*\*\*\*

-- START Invocation

-- The parent Sequence-Controlled Data Processing procedure START

-- invocation is extended with the sequContrFrameInputQueueSize parameter

-- from the 'SequContrFrameDataProcStartInvocExt extension type.

-- This extension is defined by 'StartInvocation': -- 'startInvocationExtension': 'scdpStartInvocExt':

-- 'SequContrDataProcStartInvocExt':

-- 'sequContrDataProcStartInvocExtExtension':

-- 'SequContrFrameDataProcStartInvocExt'.

-- 'StartInvocation': 'startInvocationExtension': 'scdpStartInvocExt':

-- 'SequContrDataProcStartInvocExt':

-- 'sequContrDataProcStartInvocExtExtension':

-- 'SequContrFrameDataProcStartInvocExt':

-- 'sequContrFrameDataProcStartInvocExtExtension' shall be set to -- 'notUsed'.

SequContrFrameDataProcStartInvocExt ::= SEQUENCE

{ sequContrFrameInputQueueSize CHOICE

{ unchanged [0] NULL

, modify [1] IntUnsigned

}

, sequContrFrameDataProcStartInvocExtExtension Extended

}

sequContrFrameDataProcStartInvocExt

OBJECT IDENTIFIER ::= {scfdpExtendedOpsParam 1}

-- START return

-- The START positive return does not extend StartReturn; that is,

-- 'StartReturn': 'StandardReturnHeader': 'result': 'positive' shall be set -- to 'notUsed'.

-- The START negative return does not extend StartReturn; that is, -- 'StartReturn': 'StandardReturnHeader': 'result': 'negative': -- 'negExtension' shall be set to 'notUsed'.

-- The START negative return makes use of: (a) one of the common -- diagnostics of 'StandardReturnHeader': 'result': 'negative':

-- 'diagnostic': 'Diagnostic' (see 3.3.2.7 and F3.3 of reference [1])

-- except 'diagnosticExtension'; or (b) one of the additional values -- specified by 'StartReturn': 'StandardReturnHeader': 'result': -- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'startDiagnosticExt': 'StartDiagnosticExt' in F3.4 of reference [1] -- except 'startDiagnosticExtExtension'.

-- \*\*\*\*\*

-- STOP Invocation

-- The STOP invocation is not extended; that is, 'StopInvocation': -- 'stopInvocationExtension' shall be set to 'notUsed'.

-- STOP return

-- The STOP positive return does not extend StopReturn; that is,

'StopReturn':

-- 'StandardReturnHeader': 'result': 'positive' shall be set to 'notUsed'.

CCSDS 922.3-B-0 Page C-2 December 2020

-- The STOP negative return does not extend StopReturn; that is,

'StopReturn':

-- 'StandardReturnHeader': 'result': 'negative': 'negExtension' shall be -- set to 'notUsed'.

-- The STOP negative return makes use of one of the common diagnostics -- of 'StandardReturnHeader': 'result': 'negative': 'diagnostic':

-- 'Diagnostic' (see 3.3.2.7 and F3.3 of reference [1]) except -- 'diagnosticExtension'.

-- \*\*\*\*\*

-- PROCESS-DATA Invocation

-- The parent Sequence-Controlled Frame Data Processing PROCESS-DATA

-- invocation is extended with the additional parameters from the

-- DataProcProcDataInvocExt and SequContrDataProcProcDataInvocExt extension

-- types. No further parameters are added by the Sequence-Controlled Frame

-- Data Processing procedure; that is, 'ProcessDataInvocation': -- 'processDataInvocationExtension': 'dpProcDataInvovExt':

-- 'DataProcProcDataInvocExt': 'dataProcProcDataInvocExtExtension': -- 'scdpProcDataInvocExt': 'SequContrDataProcProcDataInvocExt':

-- 'sequContrDataProcDataInvocExtExtension' shall be set to 'notUsed'.

-- PROCESS-DATA positive return

-- The parent Sequence-Controlled Data Processing PROCESS-DATA

-- positive return is extended with additional parameters from -- the SequContrDataProcProcDataPosReturnExt extension type.

-- No further parameters are added by the Sequence-Controlled Frame

-- Data Processing procedure; that is, 'ProcessDataReturn':

-- 'StandardReturnHeader': 'result': 'positive':

-- 'scdpProcDataPosReturnExt': 'SequContrDataProcProcDataPosReturnExt':

-- 'sequContrDataProcProcDataPosReturnExtExtension' shall be set to -- 'notUsed'.

-- PROCESS-DATA negative return parameters

-- The parent Sequence-Controlled Data Processing PROCESS-DATA

-- negative return is extended with additional parameters from the

-- SequContrDataProcProcDataNegReturnExt extension type.

-- No further parameters are added to the ProcessDataReturn by the

-- Sequence-Controlled Frame Data Processing procedure; that is, -- 'ProcessDataReturn': 'StandardReturnHeader': 'result': 'negative':

-- 'negExtension': 'scdpProcDataNegReturnExt': -- 'SequContrDataProcProcDataNegReturnExt':

-- 'sequContrDataProcProcDataNegReturnExtExtension'shall be set to -- 'notUsed'.

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

-- PROCESS-DATA negative return diagnostics

-- The parent Sequence-Controlled Data Processing PROCESS-DATA

-- negative return is extended with additional diagnostics from the

-- SequContrDataProcProcDataDiagnosticExt extension type. Additional

-- diagnostics are added to the ProcessDataReturn by the

-- Sequence-Controlled Frame Data Processing procedure; that is, the

-- PROCESS-DATA negative return makes use of:

-- (a) one of the common diagnostics of 'StandardReturnHeader' : 'result':

-- 'negative': 'diagnostic': 'Diagnostic' (see 3.3.2.7 and F3.3 of

-- reference [1]) except 'diagnosticExtension';

-- (b) one of the additional diagnostics defined by 'ProcessDataReturn':

-- 'StandardReturnHeader': 'result': 'negative': 'diagnostic':

-- 'Diagnostic': 'diagnosticExtension': 'scdpProcDataDiagExt':

-- 'SequContrDataProcProcDataDiagnosticExt' (see F3.10 of reference [1])

-- except 'sequContrDataProcProcDataDiagnosticExtExtension';

-- (c) 'ProcessDataReturn': 'StandardReturnHeader': 'result': 'negative':

-- 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'scdpProcDataDiagExt': 'SequContrDataProcProcDataDiagnosticExt': -- 'sequContrDataProcProcDataDiagnosticExtExtension':

-- 'SequContrFrameDataProcProcDataDiagnosticExt' except -- 'scfdpProcDataDiagnosticExtExtension'.

SequContrFrameDataProcProcDataDiagnosticExt ::= CHOICE

{ invalidGvcid [1] AdditionalText , frameTooShort [2] AdditionalText , frameTooLong [3] AdditionalText

, scfdpProcDataDiagnosticExtExtension [100] Embedded

}

sequContrFrameDataProcProcDataDiagnosticExt

OBJECT IDENTIFIER ::= {scfdpExtendedOpsParam 2}

------

-- NOTIFY invocation

-- The parent Sequence-Controlled Data Processing NOTIFY invocation

-- is extended with additional parameters from the DataProcNotifyInvocExt

-- extension type. No further parameters are added by the Sequence- -- Controlled Frame Data Processing; that is, 'NotifyInvocation':

-- 'notifyInvocationExtension': 'dpNotifyInvocExt':

-- 'DataProcNotifyInvocExt': 'dataProcNotifyInvocExtExtension' shall be set

-- to IotUsed'.

-- The parent Sequence-Controlled Data Processing NOTIFY invocation

-- extends the processing-status parameter through the

-- SequContrDataProcStatus extension type. No further processing-status

-- values are added by the Sequence-Controlled Frame Data Processing; that -- is,'NotifyInvocation': 'notifyInvocationExtension': 'dpNotifyInvocExt':

-- 'DataProcNotifyInvocExt': 'dataUnitIdLastProcessed': -- 'dataUnitLastProcessed': 'dataProcessingStatus':

-- 'dataProcessingStatusExtension': 'scdpNotifyProcStatusExt': -- 'SequContrDataProcStatus' must not be set to -- 'sequContrDataProcStatusExtension'.

-- \*\*\*\*\*

-- EXECUTE-DIRECTIVE invocation

-- The Sequence-Controlled Frame Data Processing EXECUTE-DIRECTIVE

-- invocation does not add any directive-identifiers to those already used

-- by the parent Sequence-Controlled Data Processing EXECUTE-DIRECTIVE -- (reference [1], 4.8 and F3.10).

CCSDS 922.3-B-0 Page C-4 December 2020

-- The EXECUTE-DIRECTIVE invocation is not extended; that is,

-- 'ExecuteDirectiveInvocation': 'execute DirectiveInvocationExtension' -- shall be set to 'notUsed'.

-- EXECUTE-DIRECTIVE acknowledgement

-- The EXECUTE-DIRECTIVE positive acknowledgement is not extended; that is,

-- 'ExecuteDirectiveAcknowledge': 'StandardAcknowledgeHeader':

-- 'StandardReturnHeader': 'result': 'positive' shall be set to 'notUsed'.

-- The EXECUTE-DIRECTIVE negative acknowledgement is not extended; that is,

-- 'ExecuteDirectiveAcknowledge': 'StandardAcknowledgeHeader':

-- 'StandardReturnHeader': 'result': 'negative': 'negExtension' shall be -- set to 'notUsed'.

-- The EXECUTE-DIRECTIVE negative acknowledgement makes use of: (a) one of -- the common diagnostics of 'StandardReturnHeader': 'result': 'negative':

-- 'diagnostic': 'Diagnostic' (see 3.3.2.7 of reference

-- [1]); or (b) one of

-- the additional diagnostics defined by 'ExecuteDirectiveAcknowledge': -- 'StandardAcknowledgeHeader': 'StandardReturnHeader': 'result':

-- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirAckDiagExt': 'ExecDirNegAckDiagnosticExt'. No further -- diagnostics are specified; that is, 'ExecuteDirectiveAcknowledge': -- 'StandardAcknowledgeHeader': 'StandardReturnHeader': 'result':

-- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirAckDiagExt': 'ExecDirNegAckDiagnosticExt' must not be set to -- 'execDirNegAckDiagnosticExtExtension'.

-- EXECUTE-DIRECTIVE return

-- The EXECUTE-DIRECTIVE positive return is not extended; that is,

-- 'ExecuteDirectiveReturn': 'StandardReturnHeader': 'result': 'positive' -- shall be set to 'notUsed'.

-- The EXECUTE-DIRECTIVE negative return is not extended; that is,

-- 'ExecuteDirectiveReturn': 'StandardReturnHeader': 'result': 'negative': -- 'negExtension' shall be set to 'notUsed'.

-- The EXECUTE-DIRECTIVE negative return makes use of: (a) one of the -- common diagnostics of 'StandardReturnHeader': 'result': 'negative':

-- 'diagnostic': 'Diagnostic' (see 3.3.2.7 and F3.3 of reference [1])

-- except 'diagnosticExtension'; or (b) one of the additional diagnostics -- defined by 'ExecuteDirectiveReturn': 'StandardReturnHeader': 'result':

-- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirNegReturnDiagnosticExt': 'ExecDirNegReturnDiagnosticExt'in F3.4 -- of reference [1] except 'execDirNegReturnDiagnosticExtExtension'.

END

### C2 TRANSFER SYNTAX

The SequContrFrameDataProcStartInvocExt and

SequContrFrameDataProcProcDataDiagnosticExt types specified in this module shall be encoded for transfer using the Basic Encoding Rules specified in reference [6].

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

**ANNEX D**

**PROCEDURE—BUFFERED FRAME DATA PROCESSING PDUS**

**(NORMATIVE)**

### D1 BUFFERED FRAME DATA PROCESSING PDUS ASN.1 MODULE

CCSDS-BUFFERED-FRAME-DATA-PROCESSING-PDUS

{ iso(1) identified-organization(3) standards-producing-organization(112) ccsds(4) css(4) csts(1) services(2) forwardFrame(3) forwardFrameServiceModules(4) extensions(2) bufferedFrameDataProcessingPdus(2) version (1)

}

DEFINITIONS

IMPLICIT TAGS

::= BEGIN

EXPORTS BufferedFrameDataProcessingPdu

;

IMPORTS Extended

, IntUnsigned

FROM CCSDS-CSTS-COMMON-TYPES

-- CCSDS-CSTS-COMMON-TYPES is defined in F3.3 of the -- CSTS SFW (reference [1]).

BufferedDataProcessingPdu

FROM CCSDS-CSTS-BUFFERED-DATA-PROCESSING-PDUS

-- CCSDS-CSTS-BUFFERED-DATA-PROCESSING-PDUS is defined in F3.9 -- of the CSTS SFW (reference [1]).

bfdpExtendedOpsParam

, bfdpEventsId

FROM CCSDS-FORWARD-FRAME-OBJECT-IDENTIFIERS

;

-- The BFDP procedure is derived from the CSTS

-- SFW Buffered Data Processing procedure, which in turn is derived from

-- the Data Processing procedure. Its PDU is cast as the type of the PDU

-- defined in the

-- BFDP procedure: BufferedDataProcessingPdu type

-- defined in the CCSDS-CSTS-BUFFERED-DATA-PROCESSING-PDUS module (F3.9) of -- the CSTS SFW (reference [1]).

BufferedFrameDataProcessingPdu ::= BufferedDataProcessingPdu

CCSDS 922.3-B-0 Page D-1 December 2020

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

-- \*\*\*\*\*

-- START Invocation

-- The parent Buffered Data Processing procedure START invocation is

-- extended with the buffFrameInputQueueSize,

-- buffFrameMaximumForwardBufferSize, and buffFrameProcessingLatencyLimit -- parameters from the 'BuffFrameDataProcStartInvocExt extension type.

-- This extension is defined by 'StartInvocation':

-- 'startInvocationExtension': 'buffFrameDataProcStartInvocExt':

-- 'BuffFrameDataProcStartInvocExt':

-- 'StartInvocation': 'startInvocationExtension':

-- 'buffFrameDataProcStartInvocExt ': -- 'BuffFrameDataProcStartInvocExt':

-- 'buffFrameDataProcStartInvocExtExtension' shall be set to 'notUsed'.

BuffFrameDataProcStartInvocExt ::= SEQUENCE

{ buffFrameInputQueueSize CHOICE

{ unchanged [0] NULL

, modify [1] IntUnsigned

}

, buffFrameMaximumForwardBufferSize CHOICE

{ unchanged [0] NULL

, modify [1] IntUnsigned

}

, buffFrameProcessingLatencyLimit CHOICE

{ unchanged [0] NULL

, modify [1] IntUnsigned

}

, buffFrameDataProcStartInvocExtExtension Extended

}

buffFrameDataProcStartInvocExt

OBJECT IDENTIFIER ::= {bfdpExtendedOpsParam 1}

-- START return

-- The START positive return does not extend StartReturn; that is,

-- 'StartReturn': 'StandardReturnHeader': 'result': 'positive' shall be set -- to 'notUsed'.

-- The START negative return does not extend StartReturn; that is, -- 'StartReturn': 'StandardReturnHeader': 'result': 'negative': -- 'negExtension' shall be set to 'notUsed'.

-- The START negative return makes use of: (a) one of the common -- diagnostics of 'StandardReturnHeader': 'result': 'negative':

-- 'diagnostic': 'Diagnostic' (see 3.3.2.7 and F3.3 of reference [1])

-- except 'diagnosticExtension'; or (b) one of the additional values -- specified by 'StartReturn': 'StandardReturnHeader': 'result': -- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

--'startDiagnosticExt': 'StartDiagnosticExt' in F3.4 of reference [1] -- except 'startDiagnosticExtExtension'.

-- \*\*\*\*\*

-- STOP Invocation

-- The STOP invocation is not extended; that is, 'StopInvocation': -- 'stopInvocationExtension' shall be set to 'notUsed'.

CCSDS 922.3-B-0 Page D-2 December 2020 CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

-- STOP return

-- The STOP positive return does not extend StopReturn; that is,

-- 'StopReturn':'StandardReturnHeader': 'result': 'positive' shall be set -- to 'notUsed'.

-- The STOP negative return does not extend StopReturn; that is, -- 'StopReturn':'StandardReturnHeader': 'result': 'negative': -- 'negExtension' shall be set to 'notUsed'.

-- The STOP negative return makes use of one of the common diagnostics -- of 'StandardReturnHeader': 'result': 'negative': 'diagnostic':

-- 'Diagnostic' (see 3.3.2.7 and F3.3 of reference [1]) except -- 'diagnosticExtension'.

-- \*\*\*\*\*

-- PROCESS-DATA Invocation

-- The parent Buffered Data Processing procedure PROCESS-DATA invocation is

-- extended with the additional parameter from the DataProcProcDataInvocExt

-- extension type. No further parameters are added by the Buffered Frame

-- Data Processing procedure; that is, 'ProcessDataInvocation': -- 'processDataInvocationExtension': 'dpProcDataInvocExt':

-- 'DataProcProcDataInvocExt': 'dataProcProcDataInvocExtExtension' shall be -- set to 'notUsed'.

pBFDPdataUnitId OBJECT IDENTIFIER ::= { bfdpEventsId 1}

------

-- NOTIFY invocation

-- The parent Buffered Data Processing NOTIFY invocation is extended with -- an additional parameter from the DataProcNotifyInvocExt extension type.

-- No further parameters are added by the BFDP;

-- that is, 'NotifyInvocation': 'notifyInvocationExtension':

-- 'dpNotifyInvocExt': 'DataProcNotifyInvocExt':

-- 'dataProcNotifyInvocExtExtension' shall be set to 'notUsed'.

-- The parent Buffered Data Processing NOTIFY invocation does not extend

-- the data-processing-status parameter, and no further

-- data-processing-status values are added by the Buffered Frame Data

-- Processing procedure; that is, 'NotifyInvocation': -- 'notifyInvocationExtension': 'dpNotifyInvocExt': -- 'DataProcNotifyInvocExt': 'dataUnitIdLastProcessed':

-- 'dataUnitLastProcessed': 'dataProcessingStatus' must not be set to -- 'dataProcStatusExtension'.

-- NOTIFY event-name extension

-- This procedure defines the additional eventName values 'invalid GVCID',

-- 'frame length too short', and 'frame length too long'. The associated

-- Published Identifiers are pBFDPinvalidGvcid, pBFPDframeTooShort, and -- pBFDPframeTooLong, respectively, as defined in annex F.

-- \*\*\*\*\*

END

### D2 TRANSFER SYNTAX

The BuffFrameDataProcStartInvocExt type specified in this module shall be encoded for transfer using the Basic Encoding Rules specified in reference [6].

CCSDS 922.3-B-0 Page D-3 December 2020 CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

**ANNEX E**

**PROCEDURE—MASTER THROW EVENT PDUS**

**(NORMATIVE)**

### E1 MASTER THROW EVENT PDUS ASN.1 MODULE

CCSDS-MASTER-THROW-EVENT-PDUS

{ iso(1) identified-organization(3) standards-producing-organization(112) ccsds(4) css(4) csts(1) services(2) forwardFrame(3)

forwardFrameServiceModules(4) extensions(2) masterThrowEventPdus(3) version

(1)

}

DEFINITIONS

IMPLICIT TAGS

::= BEGIN

EXPORTS MasterThrowEventPdu

;

IMPORTS AdditionalText

, Embedded

, Extended

FROM CCSDS-CSTS-COMMON-TYPES

-- CCSDS-CSTS-COMMON-TYPES is defined in F3.3 of the -- CSTS SFW (reference [1]).

ThrowEventPdu

FROM CCSDS-CSTS-THROW-EVENT-PDUS

-- FROM CCSDS-CSTS-THROW-EVENT-PDUS is defined in F3.14 of the -- CSTS SFW (reference [1]).

masterThrowEventExtendedOpsParam FROM CCSDS-FORWARD-FRAME-OBJECT-IDENTIFIERS

;

-- The Master Throw Event procedure is derived from the CSTS SFW Throw

-- Event procedure. Its PDU is cast as the type of the PDU defined in the

-- Throw Event procedure: ThrowEventPdu type is defined in the

-- CCSDS-CSTS-THROW-EVENT-PDUS module (F3.14) of the CSTS Specification -- Framework (reference [1]).

MasterThrowEventPdu ::= ThrowEventPdu

-- \*\*\*\*\*

-- EXECUTE-DIRECTIVE invocation

-- The EXECUTE-DIRECTIVE invocation is not extended; that is,

-- 'ExecuteDirectiveInvocation': 'executeDirectiveInvocationExtension' -- shall be set to 'notUsed'.

CCSDS 922.3-B-0 Page E-1 December 2020 CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

-- EXECUTE-DIRECTIVE acknowledgement

-- The EXECUTE-DIRECTIVE positive acknowledgement is not extended, i.e.,

-- 'ExecuteDirectiveAcknowledge': 'StandardAcknowledgeHeader':

-- 'StandardReturnHeader': 'result': 'positive' shall be set to 'notUsed'.

-- The EXECUTE-DIRECTIVE negative acknowledgement is not extended, i.e.,

-- 'ExecuteDirectiveAcknowledge': 'StandardAcknowledgeHeader':

-- 'StandardReturnHeader': 'result': 'negative': 'negExtension' shall be -- set to 'notUsed'.

-- The EXECUTE-DIRECTIVE negative acknowledgement diagnostics are extended

-- by the addition of the masterThroweEventDisabled diagnostic. The

-- EXECUTE-DIRECTIVE negative acknowledgement makes use of:

-- (a) one of the common diagnostics of 'StandardReturnHeader': 'result':

-- 'negative':'diagnostic': 'Diagnostic' (see 3.3.2.7 of reference [1]); or

-- (b) one of the additional diagnostics defined by

-- 'ExecuteDirectiveAcknowledge': 'StandardReturnHeader': 'result': -- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirAckAckDiagExt':'ExecDirNegAckDiagnosticExt' (see F3.4 of

-- reference [1]); or

-- (c) one of the additional diagnostics defined by

-- 'ExecuteDirectiveAcknowledge': 'StandardReturnHeader': 'result': -- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirAckAckDiagExt':'ExecDirNegAckDiagnosticExt':

-- 'execDirNegAckDiagnosticExtExtension': 'masterTeExecDirDiagExt':

-- 'MasterTeExecDirNegAckDiagnosticExt'.

-- No further diagnostics are specified, i.e.,

-- 'ExecuteDirectiveAcknowledge': 'StandardReturnHeader': 'result': -- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirAckAckDiagExt': 'ExecDirNegAckDiagnosticExt':

-- 'execDirNegAckDiagnosticExtExtension': 'masterTeExecDirDiagExt': -- 'MasterTeExecDirNegAckDiagnosticExt' must not be set to -- 'masterTeExecDirNegAckDiagnosticExtExtension'.

MasterTeExecDirNegAckDiagnosticExt ::= CHOICE

{ masterThrowEventDisabled [0] AdditionalText

, masterTeExecDirNegAckDiagnosticExtExtension [100] Embedded

}

masterTeExecDirDiagExt OBJECT IDENTIFIER ::=

{ masterThrowEventExtendedOpsParam 1}

-- EXECUTE-DIRECTIVE return

-- The EXECUTE-DIRECTIVE positive return is not extended, i.e.,

-- 'ExecuteDirectiveReturn': 'StandardReturnHeader': 'result': 'positive' -- shall be set to 'notUsed'.

-- The EXECUTE-DIRECTIVE negative return is not extended, i.e.,

-- 'ExecuteDirectiveReturn': 'StandardReturnHeader': 'result': 'negative': -- 'negExtension' shall be set to 'notUsed'.

CCSDS 922.3-B-0 Page E-2 December 2020 CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

-- The EXECUTE-DIRECTIVE negative return makes use of:

-- (a) one of the common diagnostics of 'StandardReturnHeader': 'result':

-- 'negative': 'diagnostic': 'Diagnostic' (see (3.3.2.7 and F3.3 of

-- reference [1])) except 'diagnosticExtension'; or

-- (b) one of the additional diagnostics defined by

-- 'ExecuteDirectiveReturn': 'StandardReturnHeader': 'result': -- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirNegReturnDiagnosticExt': 'ExecDirNegReturnDiagnosticExt' in F3.4

-- of reference [1] except 'execDirNegReturnDiagnosticExtExtension'; or

-- (c) one of the additional diagnostics defined by

-- 'ExecuteDirectiveReturn': 'StandardReturnHeader': 'result': 'negative':

-- 'diagnostic': 'Diagnostic': 'diagnosticExtension':

-- 'execDirNegReturnDiagnosticExt': 'ExecDirNegReturnDiagnosticExt': -- 'execDirNegReturnDiagnosticExtExtension': 'teExecDirDiagExt':

-- 'TeExecDirNegReturnDiagnosticExt' in F3.14 of reference [1] except

-- 'teExecDirNegReturnDiagnosticExtExtension'.

-- \*\*\*\*\*

END

### E2 TRANSFER SYNTAX

The MasterTeExecDirNegAckDiagnosticExt type specified in this module shall be encoded for transfer using the Basic Encoding Rules specified in reference [6].

CCSDS 922.3-B-0 Page E-3 December 2020

**ANNEX F**

**FORWARD FRAME SERVICE PROCEDURE PARAMETERS,**

**EVENTS, AND DIRECTIVES**

**(NORMATIVE)**

### F1 FORWARD FRAME SERVICE PROCEDURE PARAMETERS, EVENTS, AND DIRECTIVES ASN.1 MODULE

CCSDS-FORWARD-FRAME-SERVICE-PROCEDURE-PARAMETERS-EVENTS-DIRECTIVES { iso(1) identified-organization(3) standards-producing-organization(112) ccsds(4) css(4) csts(1) services(2) forwardFrameService(3) modules (4)

procedureParamEventDirective(3) version (1)

}

DEFINITIONS

IMPLICIT TAGS

::= BEGIN

EXPORTS PFDPauthorizedGvcidType

, PFDPgvcidBitMaskType

, PFDPmaxFrameLenType

, PFDPminFrameLenType

, PFDPnumberDataUnitsProcessedType

, PFDPnumberDataUnitsRcvdType

, PFDPnumberDataUnitsToProcessingType

, PBFDPauthorizedGvcidType

, PBFDPgvcidBitMaskType

, PBFDPmaxFrameLenType

, PBFDPminFrameLenType

, PFDPnumberDataUnitsProcessedType

, PBFDPnumberDataUnitsRcvdType

, PBFDPnumberDataUnitsToProcessingType

, PSCFDPauthorizedGvcidType

, PSCFDPgvcidBitMaskType

, PSCFDPmaxFrameLenType

, PSCFDPminFrameLenType

, PSCFDPnumberDataUnitsProcessedType

, PSCFDPnumberDataUnitsRcvdType

, PSCFDPnumberDataUnitsToProcessingType

, PMTEenableThrowEventType

;

Page F-1

IMPORTS bfdpDirectivesId

, bfdpEventsId

, bfdpExtendedOpsParam

, bfdpExtendedProcParam

, masterThrowEventDirectivesId

, masterThrowEventEventsId

, masterThrowEventExtendedOpsParam

, masterThrowEventExtendedProcParam

, scfdpDirectivesId

, scfdpEventsId

, scfdpExtendedOpsParam

, scfdpExtendedProcParam

FROM CCSDS-FORWARD-FRAME-OBJECT-IDENTIFIERS

BufferSize

, IntPos

, IntUnsigned

FROM CCSDS-CSTS-COMMON-TYPES

;

-- CCSDS-CSTS-COMMON-TYPES is defined in F3.3 of the -- CSTS Specification Framework (reference [1]).

-- ============================================================

-- Data types common to both Forward Frame Prime Procedure types

PFDPauthorizedGvcidType ::= OCTET STRING (SIZE (4)) PFDPgvcidBitMaskType ::= OCTET STRING (SIZE (4))

PFDPmaxFrameLenType ::= IntPos

PFDPminFrameLenType ::= IntPos

PFDPnumberDataUnitsRcvdType ::= IntUnsigned

PFDPnumberDataUnitsToProcessingType ::= IntUnsigned PFDPnumberDataUnitsProcessedType ::= IntUnsigned

-- SEQUENCE-CONTROLLED FRAME DATA PROCESSING derivations of the common data

-- types

pSCFDPauthorizedGvcid OBJECT IDENTIFIER

::= {scfdpExtendedProcParam 1}

PSCFDPauthorizedGvcidType ::= PFDPauthorizedGvcidType

pSCFDPgvcidBitMask OBJECT IDENTIFIER

::= {scfdpExtendedProcParam 2}

PSCFDPgvcidBitMaskType ::= PFDPgvcidBitMaskType

pSCFDPmaxFrameLen OBJECT IDENTIFIER

::= {scfdpExtendedProcParam 3}

PSCFDPmaxFrameLenType ::= PFDPmaxFrameLenType

pSCFDPminFrameLen OBJECT IDENTIFIER

::= {scfdpExtendedProcParam 4}

PSCFDPminFrameLenType ::= PFDPminFrameLenType

pSCFDPnumberDataUnitsRcvd OBJECT IDENTIFIER

::= {scfdpExtendedProcParam 5}

PSCFDPnumberDataUnitsRcvdType ::= PFDPnumberDataUnitsRcvdType

Page F-2

pSCFDPnumberDataUnitsToProcessing OBJECT IDENTIFIER

::= {scfdpExtendedProcParam 6}

PSCFDPnumberDataUnitsToProcessingType ::=

PFDPnumberDataUnitsToProcessingType

pSCFDPnumberDataUnitsProcessed OBJECT IDENTIFIER

::= {scfdpExtendedProcParam 7}

PSCFDPnumberDataUnitsProcessedType ::=

PFDPnumberDataUnitsProcessedType

-- BUFFERED FRAME DATA PROCESSING derivations of the common data types

pBFDPauthorizedGvcid OBJECT IDENTIFIER

::= {bfdpExtendedProcParam 1}

PBFDPauthorizedGvcidType ::= PFDPauthorizedGvcidType

pBFDPgvcidBitMask OBJECT IDENTIFIER

::= {bfdpExtendedProcParam 2}

PBFDPgvcidBitMaskType ::= PFDPgvcidBitMaskType

pBFDPmaxFrameLen OBJECT IDENTIFIER

::= {bfdpExtendedProcParam 3}

PBFDPmaxFrameLenType ::= PFDPmaxFrameLenType pBFDPminFrameLen OBJECT IDENTIFIER

::= {bfdpExtendedProcParam 4}

PBFDPminFrameLenType ::= PFDPminFrameLenType

pBFDPnumberDataUnitsRcvd OBJECT IDENTIFIER

::= {bfdpExtendedProcParam 5}

PBFDPnumberDataUnitsRcvdType ::= PFDPnumberDataUnitsRcvdType

pBFDPnumberDataUnitsToProcessing OBJECT IDENTIFIER

::= {bfdpExtendedProcParam 6}

PBFDPnumberDataUnitsToProcessingType ::=

PFDPnumberDataUnitsToProcessingType

pBFDPnumberDataUnitsProcessed OBJECT IDENTIFIER

::= {bfdpExtendedProcParam 7}

PBFDPnumberDataUnitsProcessedType ::=

PFDPnumberDataUnitsProcessedType

pBFDPframeTooLong OBJECT IDENTIFIER ::= {bfdpEventsId 1} pBFDPframeTooShort OBJECT IDENTIFIER ::= {bfdpEventsId 2} pBFDPinvalidGvcid OBJECT IDENTIFIER ::= {bfdpEventsId 3}

-- MASTER THROW EVENT

pMTEenableThrowEvent OBJECT IDENTIFIER ::=

{masterThrowEventExtendedProcParam 1}

PMTEenableThrowEventType ::= ENUMERATED

{ enabled (0) , disabled (1)

}

END

Page F-3

### F2 TRANSFER SYNTAX

The PFDPgvcidBitMaskType, PFDPauthorizedGvcidType,

PFDPminFrameLenType, PFDPmaxFrameLenType,

PFDPNumberDataUnitsRcvdType,

PFDPNumberDataUnitsToProcessingType,

PFDPNumberDataUnitsProcessedType, and PMTEenableThrowEventType

types specified in this module shall be encoded for transfer using the Basic Encoding Rules specified in reference [6].

Page F-4

**ANNEX G**

**FORWARD FRAME SERVICE PRODUCTION REQUIREMENTS**

**(NORMATIVE)**

### G1 GENERAL

Overall, the requirements on the production processes that underlie the Forward Frame service are standard functions that are defined in various CCSDS carrier modulation, synchronization and channel coding, and space data link protocol Recommended Standards.

As described and illustrated in section 2, the Forward Frame service does not use a single combination of these carrier modulation, synchronization and channel coding, and SDLP standards. Rather, the Forward Frame service is intended to work with a variety of these standards in the various combinations in which they can be legally configured. It is therefore not viable to specify a single Forward Frame Service Production process. Instead, it is appropriate to identify the requirements that the Forward Frame service levies on all production processes.

The resources that constitute the production processes that perform both their normative functions (as defined by the appropriate CCSDS Recommended Standards) as well as these Forward Frame-specific requirements are necessary to the successful operation of the Forward Frame service but are not part of the Forward Frame transfer service. The Forward Frame transfer service instances and the production resources that support them are configured by Service Management to operate together.

As described in 2.2, there are two types of production configuration: the Transfer Frame configuration and the CADU configuration. The following subsections specify the requirements that the FF service levies on the underlying production processes in each of these configuration types.

NOTE – These requirements do not include any extended requirements that other CSTSes or SLE transfer services may levy on these production functions.

### G2 TRANSFER FRAME CONFIGURATION G2.1 GENERAL

The production for the Transfer Frame configuration is illustrated in figure 2-1, and consists of functions performed in the SDLP, Synchronization and Channel Coding, Physical Channel, and Aperture functional strata.

Subsection I2 provides an example configuration of the Transfer Frame configuration using TC Transfer Frames. Subsection I3 provides an example configuration of the Transfer Frame configuration using AOS Transfer Frames. Subsection I4 provides an example configuration of the Transfer Frame configuration using variable-length USLP Transfer Frames. Subsection I5 provides an example configuration of the Transfer Frame configuration using variable-length Transfer Frames.

### G2.2 SPACE DATA LINK PROTOCOL

#### G2.2.1 Virtual Channel Multiplexing

**G2.2.1.1** The production process shall multiplex frames from different Virtual Channels into a single MC, in conformance with the requirements specified in the published CCSDS Recommended Standard for the applicable SDLP.

NOTE – References [L12] (Telecommand), [L13] (AOS), and [L14] (USLP) are the SDLPs that are explicitly supported by this Recommended Standard.

**G2.2.1.2** Each instance of the VC Multiplexing function shall support one of three VC multiplexing schemes: absolute priority, polling vector, and First-In-First-Out (FIFO), as configured by Service Management.

NOTE – In the following subparagraphs, the descriptions of the multiplexing schemes are conceptual and do not levy any design constraints on actual implementations, as long as the resultant behavior occurs.

**G2.2.1.2.1** In the absolute priority multiplexing scheme, a configurable vector specifies the priority of each VC that is handled by the VC Multiplexing function. The VC Multiplexing function contains a virtual queue for each VC that it handles. Frames are time tagged upon receipt by the VC Multiplexing function and placed into their respective VC queues in order of time of receipt. The VC Multiplexing function instance makes available to the associated MC Multiplexing function (a) whether it currently has any frames in any of its queues and (b) the time tag of the first frame in the queue of the highest-priority VC that contains a frame. When the MC Multiplexing function gives its release token to the VC Multiplexing function (see G2.2.2.2), if any of the VC queues has any frames, the VC Multiplexing function shall release the first frame in the queue of the highest-priority VC that contains a frame.

**G2.2.1.2.2** In the polling vector scheme, a configurable vector specifies the sequence in which the VCs that are handled by the VC Multiplexing function are to be polled. The VC Multiplexing function contains a virtual queue for each VC that it handles. Frames are time tagged upon receipt by the VC Multiplexing function and placed into their respective VC queues in order of time of receipt. The VC Multiplexing function instance makes available to the associated MC Multiplexing function (a) whether it currently has any frames in any of its queues and (b) the time tag of the first frame in the queue of the next VC in the polling cycle that contains a frame. When the MC Multiplexing function gives its release token to the VC Multiplexing function (see G2.2.2.2), if any of the VC queues has any frames, the VC Multiplexing function shall release the first frame of the queue of the next VC in the polling sequence that has a frame. The VC Multiplexing function shall retain the position in the polling sequence from which the frame was released so that the next time that the release token is made available to the VC Multiplexing function it will resume polling at the next VC in the polling sequence.

**G2.2.1.2.3** In the FIFO multiplexing scheme, frames are time tagged upon receipt by the VC Multiplexing function and placed into single virtual regardless of their VCs, in order of time of receipt. The VC Multiplexing function instance makes available to the associated MC Multiplexing function (a) whether it currently has any frames in its queue and (b) the time tag of the first frame in the queue. When the MC multiplexing function gives its release token to the VC Multiplexing function, if the VC Multiplexing function has any frames in its queue, it shall release the frame at the front of its queue.

**G2.2.1.3** Each instance of the VC Multiplexing function shall be capable of receiving, multiplexing, and sending Transfer Frames that are annotated with the values of the dataunit-id of the source PROCESS-DATA invocation, the procedure-instanceidentifier of the source (Sequence-Controlled or Buffered) Frame DP procedure instance, and the service-instance-identifier of the source Forward Frame service instance (see 4.6.7 of reference [1]).

**G2.2.1.4** Each resource that performs the VC Multiplexing function shall make available its current resource status: ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’.

**G2.2.1.5** Upon each change in resource status, each resource that performs the VC Multiplexing function shall emit a ‘resource status change’ event identifying the new resource status.

**G2.2.1.6** Upon being requested to discard data units, each resource that performs the VC Multiplexing function shall discard all frames that are tagged with the serviceinstance-identifier value that is specified in the ‘discard data units’ request (see 4.6.7 of reference [1]).

#### G2.2.2 Master Channel Multiplexing

**G2.2.2.1** The production process shall multiplex frames from different MCs into a single physical channel frame stream, in conformance with the requirements specified in the published CCSDS Recommended Standards for the applicable SDLP.

NOTE – References [L12] (Telecommand), [L13] (AOS), and [L14] (USLP) are the SDLPs that are explicitly supported by this Recommended Standard.

**G2.2.2.2** Each instance of the MC Multiplexing function shall support one of three VC multiplexing schemes: absolute priority, polling vector, or FIFO, as configured by Service Management.

NOTE – In the following subparagraphs, the descriptions of the multiplexing schemes are conceptual and do not levy any design constraints on actual implementations, as long as the resultant behavior occurs.

**G2.2.2.2.1** In the absolute priority multiplexing scheme, a configurable vector specifies the priority of each MC that is handled by the MC Multiplexing function. When the VC Multiplexing function needs to select the next frame to be released, the MC Multiplexing function scans the VC Multiplexing functions associated with it to determine which of the VC Multiplexing functions have a frame ready for release. The VC Multiplexing function gives the release token to the VC Multiplexing function instance that handles the highestpriority MC (according to the priority vector) that has a frame ready for release, and receives the frame from that VC Multiplexing function instance.

**G2.2.2.2.2** In the polling vector scheme, a configurable vector specifies the sequence in which the MCs that are handled by the MC Multiplexing function are to be polled. When the VC Multiplexing function needs to select the next frame to be released, the MC Multiplexing function scans the VC Multiplexing functions associated with it to determine which of the VC Multiplexing functions have a frame ready for release. The VC Multiplexing function gives the release token to the VC Multiplexing function instance that handles the next MC in the polling cycle (according to the polling vector) that has a frame ready for release, and receives the frame from that VC Multiplexing function instance. The MC Multiplexing function shall retain the position in the polling sequence from which the frame was released so that the next time that a transmission timeslot is made available to the MC Multiplexing function it will resume polling at the next MC in the polling sequence.

**G2.2.2.2.3** In the FIFO multiplexing scheme, when the VC Multiplexing function needs to select the next frame to be released, the MC Multiplexing function scans the VC Multiplexing functions associated with it to determine which of the VC Multiplexing functions have a frame ready for release. The VC Multiplexing function gives the release token to the VC Multiplexing function instance that that has the frame with the earliest time tag, and receives the frame from that VC Multiplexing function instance.

**G2.2.2.3** Each instance of the MC Multiplexing function shall be capable of receiving, multiplexing, and sending Transfer Frames that are annotated with the values of the dataunit-id of the source PROCESS-DATA invocation, the procedure-instanceidentifier of the source (Sequence-Controlled or Buffered) Frame DP procedure instance, and the service-instance-identifier of the source Forward Frame service instance (see 4.6.7 of reference [1]).

**G2.2.2.4** Each resource that performs the MC Multiplexing function shall make available its current resource status: ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’.

**G2.2.2.5** Upon each change in resource status, each resource that performs the MC Multiplexing function emits a ‘resource status change’ event identifying the new resource status.

**G2.2.2.6** Upon being requested to discard data units, each resource that performs the MC Multiplexing function shall discard all frames that are tagged with the serviceinstance-identifier value that is specified in the ‘discard data units’ request (see 4.6.7 of reference [1]).

#### G2.2.3 Only Idle Data Frame Generation

**G2.2.3.1** The production process shall generate Only Idle Data Frames as necessary to preserve the continuity of the transmitted stream of fixed-length transfer frames in the event that there are no valid Transfer Frames available for transmission at a release time.

**G2.2.3.2** Each resource that performs the Only Idle Data Frame Generation function shall make available its current resource status: ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’.

**G2.2.3.3** Upon each change in resource status, each resource that performs the Only Idle Data Frame Generation function emits a ‘resource status change’ event identifying the new resource status.

**G2.2.3.4** Upon being requested to discard data units, each resource that performs the Only Idle Data Frame Generation function shall discard all frames that are tagged with the service-instance-identifier value that is specified in the ‘discard data units’ request (see 4.6.7 of reference [1]).

### G2.3 SYNCHRONIZATION AND CHANNEL CODING

**G2.3.1** The production process shall synchronize and encode frames from the source physical channel data unit stream to form a single physical channel symbol stream, in conformance with the requirements specified in the published CCSDS Recommended Standards for synchronization and channel coding.

NOTE – Reference [L11] is the CCSDS Recommended Standard for synchronization and channel coding of variable-length frames on a forward link. As of the publication of this Recommended Standard, a planned revised version of reference [L15] will specify the synchronization and channel coding functions for fixed-length frames on the forward link.

**G2.3.2** Each resource that performs the Synchronization and Channel Coding functions shall be capable of receiving frames or groups of frames that are annotated with the values of the data-unit-id parameter(s) of the source PROCESS-DATA invocation(s), the procedure-instance-identifier of the source (Sequence-Controlled or Buffered) Frame DP procedure instance, and the service-instance-identifier of the source Forward Frame service instance (see 4.6.7 of reference [1]).

**G2.3.3** Each resource that performs the Synchronization and Channel Coding functions shall maintain the association of each frame with its data-unit-id and serviceinstance-id annotation until all encoding and/or synchronization functions have been performed for that frame.

**G2.3.4** Each resource that performs the Synchronization and Channel Coding functions shall emit a ‘data unit processing completed’ event at the detected or estimated radiation time of each frame. Each ‘data unit processing completed’ event shall carry the data-unit-id, procedure-instance-identifier, and service-instance-id annotation that accompanied the radiated frame.

**G2.3.5** Each resource that performs the Synchronization and Channel Coding functions shall make available its current resource status: ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’.

**G2.3.6** Upon each change in resource status, each resource that performs the Synchronization and Channel Coding functions shall emit a ‘resource status change’ event identifying the new resource status.

**G2.3.7** Upon being requested to discard data units, each resource that performs the Synchronization and Channel Coding functions shall discard all frames that are tagged with the -instance-identifier value that is specified in the ‘discard data units’ upon being requested to discard data units (see 4.6.7 of reference [1]).

### G2.4 PHYSICAL CHANNEL

**G2.4.1** The production process shall modulate the physical channel symbol stream for radiation across the space link, in conformance with the requirements specified in the published CCSDS Recommended Standards for forward link carrier modulation.

NOTE – At of the publication of this Recommended Standard, the set of published CCSDS Recommended Standards for forward link carrier modulation consists of references [L9] and [L10].

**G2.4.2** Each resource that performs the Physical Channel functions shall make available its current resource status: ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’.

**G2.4.3** Upon each change in resource status, each resource that performs the Physical Channel functions shall emit a ‘resource status change’ event identifying the new resource status.

### G2.5 APERTURE

**G2.5.1** The production process shall radiate the modulated carrier signal through an aperture that is appropriate for that carrier signal.

NOTE – There are no CCSDS standards for apertures.

**G2.5.2** Each aperture shall make available its current resource status: ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’.

**G2.5.3** Upon each change in resource status, each aperture shall emit a ‘resource status change’ event identifying the new resource status.

### G3 CADU CONFIGURATION G3.1 GENERAL

The production for the CADU configuration is illustrated in figure 2-2, and consists of functions performed in the Synchronization and Channel Coding, Physical Channel, and Aperture functional strata.

Subsection I6 provides an example of the CADU configuration.

The Forward Frame service requirements on the underlying Physical Channel and Aperture production processes in the CADU configuration are the same as those in the Transfer Frame configuration (see G2.4 and G2.5, respectively). Only the Synchronization and Channel Coding are specific to the CADU configuration.

### G3.2 SYNCHRONIZATION AND CHANNEL CODING

**G3.2.1** The production process shall generate Only Idle Data CADUs as necessary to preserve the continuity of the transmitted stream in the even that there are no valid Transfer Frames available for transmission at a release time, and multiplex the Only Idle Data CADUs with the CADUs received via the Forward Frame service instance. The requirement for the Only Idle Data Generation and Multiplexing function is specific to the Forward Frame service, and the details are defined in the following subparagraphs.

**G3.2.1.1** Each instance of the Only Idle Data CADU Generation and Multiplexing function shall be configured by Service Management through an Only Idle Data CADU configuration parameter. The Only Idle Data CADU shall be formed by starting with the Only Idle Data Frames for the appropriate SDLP (as specified in reference [L13] for AOS and reference [L14] for fixed-length USLP), and applying the same synchronization and channel coding transformations as those that are applied by the user of the Forward Frame service instance user:

1. If the Forward Frame service instance user has applied Reed-Solomon or LDPC encoding to the frames, the same encoding shall have been applied to the Only Idle Data Frame;
2. If the Forward Frame service instance user has applied randomization to the frames, the same randomization shall have been applied to the (unencoded or encoded) Only Idle Data Frame; and
3. The resulting unencoded, encoded and/or randomized Only Idle Data Frame shall have the Attached Synchronization Marker that is appropriate to the applied coding scheme attached.

**G3.2.1.2** Each resource that performs the Only Idle Data CADU Generation and Multiplexing function shall be capable of receiving CADUs that are annotated with the values of the data-unit-id parameter of the source PROCESS-DATA invocation, the procedure-instance-identifier of the source Frame DP procedure instance, and the service-instance-identifier of the source Forward Frame service instance

(see 4.6.7 of reference [1]).

**G3.2.1.3** Each resource that performs the Only Idle Data CADU Generation and Multiplexing function shall make available its current resource status: ‘configured’, ‘operational’, ‘interrupted’, or ‘halted’.

**G3.2.1.4** Upon each change in resource status, each resource that performs the Only Idle Data CADU Generation and Multiplexing function emits a ‘resource status change’ event identifying the new resource status.

**G3.2.1.5** Upon being requested to discard data units, each resource that performs the Only Idle Data CADU Generation and Multiplexing function shall discard all frames that are tagged with the service-instance-identifier value that is specified in the

‘discard data units’ request (see 4.6.7 of reference [1]).

**ANNEX H**

**SECURITY, SANA, AND PATENT CONSIDERATIONS**

**(INFORMATIVE)**

### H1 SECURITY CONSIDERATIONS H1.1 INTRODUCTION

This subsection describes security aspects of the Forward Frame service.

The CSTS SFW (reference [1]) explicitly provides authentication and access control for CSTSes. As one of the suite of CSTSes, the Forward Frame service inherits the authentication and access control capabilities defined in the CSTS SFW. The Forward Frame service provides no service-specific security capabilities. As specified in the CSTS SFW, additional security capabilities, if required, are levied on the underlying communications services that support the FF-CSTS. Specification of the various underlying communications technologies, and in particular their associated security provisions, are outside the scope of this Recommended Standard.

### H1.2 SECURITY CONCERNS WITH RESPECT TO THE FORWARD FRAME SERVICE

The Statements of Security Concerns subsection of the CSTS SFW (H1 of reference [1]) identifies the support for capabilities that respond to security concerns in the areas of data privacy (also known as confidentiality), data integrity, authentication, access control, availability of resources, and auditing.

### H1.3 POTENTIAL THREATS AND ATTACK SCENARIOS

#### H1.3.1 Breach of Privacy

As a member of the suite of CSTSes, the Forward Frame service depends on unspecified security mechanisms operating in the underlying communications service, the use of the CCSDS Space Data Link Security (SDLS) protocol (reference [L20]) to secure the Transfer Frames and CADUs transferred by the Forward Frame service, and/or privacy-ensuring capabilities in the service-specific application processes that interoperate through the Framework procedures, to ensure data privacy (confidentiality). If no such mechanisms are actually implemented, or if the mechanisms selected are inadequate or inappropriate to the network environment in which the mission is operating, an attacker could read the data contained in the Transfer Frames or CADUs as they traverse the wide area network from the service user to the service provider.

Page H-1

#### H1.3.2 PDU Interception and Replay

The CSTS SFW Recommended Standard (reference [1]) constrains the ability of a third party to seize control of an active CSTS instance. In addition, when operating in the sequencecontrolled data processing mode (see 1.6.1.12.2), the Forward Frame service strictly enforces sequencing of data units, such that the repetition of data unit sequence numbers (as would occur if an attacker were to attempt to inject Transfer Frames into an active service association) would cause that service association to be aborted.

Beyond those capabilities, the Forward Frame service does not specify mechanisms that would prevent an attacker from intercepting the PDUs and replacing the payload data (in this case Transfer Frames or CADUs) of those PDUs. Successful interception and replacement could allow the attacker to seize control of the spacecraft and/or its payload(s). Other than the sequencing enforcement performed when operating in the sequence-controlled data processing mode, the prevention of interception and replay attacks depends on unspecified mechanisms in the underlying communications service and/or the use of SDLS (reference [L20]). If no such mechanisms are actually implemented, or the mechanisms selected are inadequate or inappropriate to the network environment in which the mission is operating, an attacker could inject Transfer Frames transferred between the service user and the service provider without detection.

#### H1.3.3 Unauthenticated Access

The Forward Frame service uses the optional CSTS authentication capability specified in 3.2.4 of reference [1]. Beyond that authentication, each Forward Frame service instance is configured to accept the Transfer Frames for only a single designated VC and frame size constraints, rejecting any Transfer Frames that do not have the GVCID or proper frame size for that service instance. If the CSTS authentication capability is not used, and if authentication is not ensured by the underlying communications service, an attacker could somehow obtain the valid initiator-identifier values, associated GVCID, and frame size constraints for a given Forward Frame service instance and use them to bind to and activate that Forward Frame service instance and thereby inject Transfer Frames or CADUs into the forward space link associated with that service instance.

#### H1.3.4 Denial of Service Attacks

The Forward Frame service depends on unspecified mechanisms operating in the underlying communications service to ensure that the supporting network has sufficient resources to provide sufficient support to legitimate service users. If no such mechanisms are actually implemented, or the mechanisms selected are inadequate or inappropriate to the network environment in which the mission is operating, an attacker could prevent legitimate service users from using the Forward Frame service.

Page H-2

#### H1.3.5 Failure to Detect Breach Attempts

The Forward Frame service authenticates attempts to bind to it, accepts only those inputs that meet VC and Transfer Frame size constraints, and enforces the protocol on the interface with the service user. Errors in any of these areas could indicate a breach attempt. However, the Forward Frame service contains no specific capabilities to log and/or analyze such errors. Any such security auditing capability must be provided by the service provider that implements the Forward Frame service. If the service provider of the Forward Frame service provides no security auditing capabilities, or if a service user chooses not to employ auditing capabilities that do exist, then attackers may delay or escape detection while probing the service to determine the right combination of credentials and service parameters needed to successfully access the service.

### H2 SANA CONSIDERATIONS H2.1 CCSDS CSTS SERVICES REGISTRY

As described in H2.3 of reference [1], SANA has established the CSTS services subtree for the registration of OIDs for CSTS types and their associated service-specific parameters, procedures, and ASN.1 modules. This Recommended Standard adds the OIDs and labels contained in annexes B, C, D, E, and F to the CCSDS OID registry CSTS services subtree.

### H2.2 FUNCTIONAL RESOURCES REGISTRY

The FF-CSTS relies on the SANA Functional Resources registry (reference [4]) to provide the identification and definition of FR parameters and events.

As described in this Recommended Standard, the FF-CSTS configures and reports parameters and notifies events that are named in the context of FRs. Functional Resource Types are registered under the

{iso identified-organization (3) standards-producing-organization (112) ccsds (4) css (4) crossSupportResources (2)} node of the OID registration tree.

There are two subnodes under the crossSupportResources node: crossSupportFunctionalities and agencyFunctionalities, used to register CCSDS-standard Functional Resource Types and agency-unique Functional Resource Types, respectively. Under each Functional Resource Type OID, the parameters, events, and directives are registered under dedicated subnodes.

The FF-CSTS relies on the SANA Functional Resource registry to register the following:

* the FF-CSTS Provider Functional Resource Type, registered under the crossSupportFunctionalities with the classifier ffCstsProvider;

Page H-3

* the parametersId node of the ffCstsProvider subtree as {ffCstsProvider 1};
* the eventsId node of the ffCstsProvider subtree as {ffCstsProvider 2};
* the directivesId node of the ffCstsProvider subtree as {ffCstsProvider 3}.

Maintenance of the SANA registry of the FR types, parameters, events, and directives under the crossSupportFunctionalities subnode is under the purview of the CCSDS Cross Support Services Area in accordance with the process and procedures identified in the CSTS SFW.

Maintenance of the SANA registry of the FR types, parameters, events, and directives under the agencyFunctionalities subnode is under the purview of designated Agency-level control authorities. The process and procedure for designating Agency-level control authorities is documented in the CSTS SFW.

### H3 PATENT CONSIDERATIONS

There are no patents that are known to apply to the technology used in the Forward Frame service.

Page H-4

**ANNEX I**

## EXAMPLE FORWARD FRAME PROVISION AND PRODUCTION

**CONFIGURATIONS**

**(INFORMATIVE)**

### I1 GENERAL

As described in other sections of this Recommended Standard, the Forward Frame service is designed to transfer Telecommand transfer frames (as defined in reference [L12]), AOS transfer frames (as defined in reference [L13]), variable-length and fixed length USLP transfer frames (as defined in reference [L14]), and CADUs (as defined in reference [L15]).

This annex illustrates each of these five service configurations as compositions of the CCSDS-standard space data link protocol, synchronization and channel coding, and radio frequency modulation functions that are applicable to each service configuration. For each Forward Frame service configuration, the relevant layers of the CCSDS protocol stack and the CCSDS Recommended Standards that specify the functions for each of those layers as appropriate to the given service configuration are identified.

All five of these service configurations can operate with the Physical Layer functionality specified by either the *Radio Frequency and Modulation Systems—Part 1: Earth Stations and Spacecraft* Recommended Standard (CCSDS 401.0, reference [L9]) or the *Data Transmission and PN Ranging for 2 GHz CDMA Link via Data Relay Satellite* Recommended Standard (CCSDS 415.1, reference [L10]). These Physical Layer functions are not further addressed in the following subsections, although they are included in the diagrams depicting each of the service configurations.

NOTE – The CCSDS Recommended Standards identified in this annex are the ones that are relevant to the five Forward Frame service configurations as of the publication of this Recommended Standard. Future Recommended Standards that define functions that conform to the CCSDS space link layers may also be able to be used in conjunction with the Forward Frame service, but validation of such use is outside the scope of this Recommended Standard.

Each diagram depicting one of the service configurations also includes an Antenna element. The Forward Frame service has no dependency on any characteristics of antenna (or any aperture). The only requirements are that the antenna be suitable for the FR modulation and frequency bands used by the associated Physical Layer functions. The Antenna elements are included in the diagrams only to illustrate all of the functions that are included in Service Package that provides Forward Frame service.

### I2 TRANSFER FRAME CONFIGURATION FOR TELECOMMAND FRAMES I2.1 GENERAL

Figure I-1 depicts the functions that comprise the provision and production of the Forward Frame service when configured to support the transfer and radiation of TC frames. These functions correspond to the operational scenario described in 2.5.2.

NOTE – Figure 2-5 shows a ‘CLCWs from return link VC’ arrow into the TC

Synchronization and Channel Coding functions that does not appear in figure I-1. As discussed in 2.5.2 the values of the CLCW flags are used to determine the resource status of the functional resource that represents the TC Synchronization and Channel Coding functions but does not directly affect the TC Synchronization and Channel Coding functions themselves. The CLCW feedback is outside the focus of this subsection.

### I2.2 FORWARD FRAME CSTS PROVIDER

The FF-CSTS Provider represents an instance of an entity that performs the functions that are specified in this Recommended Standard. In the Transfer Frame configuration, multiple instances of the FF-CSTS Provider may exist for one space link physical channel. Each FFCSTS Provider supplies the frames from one virtual channel. In the TC Frames configuration, the FF-CSTS Provider bit mask is configured to match data units for which the TFVN, SCID, and VCID fields in the first four octets of the TC Transfer Frame Primary Header correspond to the GVCID of the virtual channel assigned to that transfer service instance.

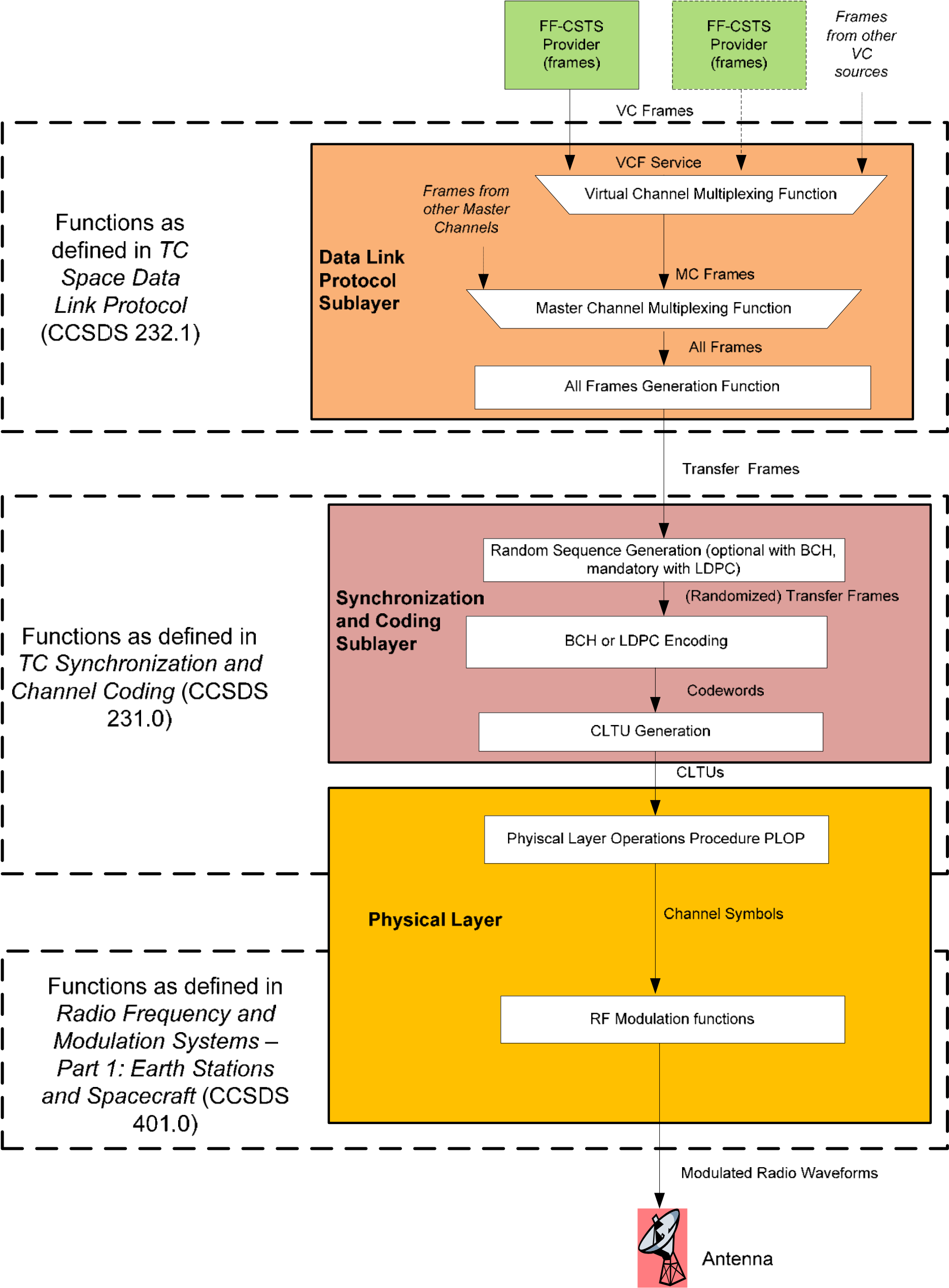
### I2.3 DATA LINK PROTOCOL SUBLAYER

#### I2.3.1 General

For the TC Frames configuration, the Data Link Protocol Sublayer functions are defined in the *TC Space Data Link Protocol* Recommended Standard (reference [L12]). The functions of this sublayer that are relevant to the transfer of TC Frames are the Virtual Channel Multiplexing Function, the Master Channel Multiplexing Functions, and the All Frames Generation Function.

#### I2.3.2 Virtual Channel Multiplexing Function

The Virtual Channel Multiplexing function of reference [L12] multiplexes the Transfer Frames from one or more VCs with the same MCID into a single stream of Transfer Frames. The sources of these VCs can be multiple FF-CSTS service instances, VC frames generated by the TC SDLP’s Virtual Channel Generation Function (see reference [L12]) and/or other VC-generating sources of externally-generated transfer frames input via the TC SDLP’s VC Frame Service (see reference [L12]).



**Figure I-1: Forward Frame Service—Transfer Frame Configuration—TC Frames** The specification of the Virtual Channel Multiplexing function in reference [L12] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component VCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. When used with the Forward Frame service the Virtual Channel Multiplexing function supports the three multiplexing schemes (absolute priority, polling vector, and FIFO) that G2.2.1.2 specifies must be implemented by any instance of the Virtual Channel Multiplexing function that receives its frames from an instance of the Forward Frame service.

#### I2.3.3 Master Channel Multiplexing Function

The Master Channel Multiplexing function of reference [L12] multiplexes the Transfer Frames from one or more Master Channels into a single stream of Transfer Frames destined for the space Physical Channel. The sources of these Master Channels can be one or more Virtual Channel Multiplexing function instances and/or other VC-generating sources of externally-generated transfer frames input via the TC SDLP’s MC Frame Service (see reference [L12]).

The specification of the Master Channel Multiplexing function in reference [L12] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component MCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. When used with the Forward Frame service the Master Channel Multiplexing function supports the three multiplexing schemes (absolute priority, polling vector, and FIFO) that G2.2.2.2 specifies must be implemented by any instance of the Master Channel Multiplexing function that receives frames from any VC Multiplexing function instance used by a Forward Frame service instance.

#### I2.3.4 All Frames Generation Function

The All Frames Generation function of reference [L12] optionally adds an FECF to the trailer of each frame and groups Transfer Frames together into a data unit that will be transformed by the TC Synchronization and Channel Coding sublayer (see reference [L11]) into a single CLTU. How many frames are placed in any individual data unit is based on configuration parameters set by Service Management that specify the maximum allowed number of frames in a data unit and the maximum allowed size (in octets) of a data unit.

When being submitted to the underlying Synchronization and Channel Coding sublayer (I2.4), the same data unit may be submitted multiple times. The number of times a data unit is submitted is based on configuration parameters set by Service Management that specify how many times Sequence-Controlled (type AD) and COP Control Command (type BC) Telecommand frames for each virtual channel are to be repeated, and whether the specific group-of-frames data unit contains any AD or BC frames that are specified to be repeated (Expedited (type BD) frames are not repeated; see reference [L12]).

NOTE – As specified in references [L11] and [L12], the Data Link Protocol Sublayer functions include the capability to cause the Synchronization and Channel Coding Sublayer (I2.4) to repeat AD and BC frames within CLTUs. The mechanism for accomplishing this is formally specified in references [L11] as a ‘data unit’ consisting of a group of frames to be transmitted in a single CLTU, accompanied by a *Repetitions* parameter that specifies the number of times that the resultant CLTU is to be transmitted. The same effect on the space link can be accomplished by having the Data Link Protocol Sublayer submit the group-offrames data unit to the Synchronization and Channel Coding Sublayer *Repetitions* times. Having the group-of-frames submitted to the Sync and Channel Coding Sublayer *Repetitions* times is the way that the repetition capability is implemented for SLE Forward CLTU (reference [L4]) and Forward Space Packet (FSP - reference [L19]): for the F-CLTU service the service user transfers the same (already formed) CLTU *Repetitions* times, and for the FSP service the Data Link Protocol Sublayer forms and submits the group-of-frames data unit to the Synchronization and Channel Coding Sublayer *Repetitions* times (based on Service Management configuration parameters). For consistency with those SLE implementations and in recognition that Forward Frame service instances may share space links with FSP service instances, the description of the repetition capability in this informative annex is phrased in terms of the approach in which the Data Link Protocol Sublayer submits the group-of-frames data unit to the Synchronization and Channel Coding Sublayer *Repetitions* times.

### I2.4 SYNCHRONIZATION AND CHANNEL CODING SUBLAYER

For the TC frames configuration, the Synchronization and Channel Coding sublayer functions are defined in the *TC Synchronization and Channel Coding* Recommended Standard (reference [L11]). The functions of reference [L11] that are relevant to the Forward Frame service are:

1. the Random Sequence Generation function, which is optional when BCH encoding is applied but mandatory when LDPC encoding is applied;
2. either the modified BCH Coding function (as specified reference [L11]) or the LDPC Coding function; and
3. the CLTU Generation function.

Reference [L11] also specifies two Physical Layer Operations Procedures (PLOPs), PLOP-1 and PLOP-2. PLOP-1 is deprecated, and only PLOP-2 is permitted to be used by missions whose planning began after September 2010. However, since the Forward Frame service is a cross support service of ESLTs that can reasonably be expected to support legacy missions, the production functions supporting Forward Frame service instances that transfer TC frames are expected to implement both PLOP-1 and PLOP-2.

NOTE – Although the definitions of the PLOPs are specified in reference [L11], they are formally considered to be functions of the Physical Layer.

### I3 TRANSFER FRAME CONFIGURATION FOR AOS FRAMES I3.1 GENERAL

Figure I-2 depicts the functions that comprise the provision and production of the Forward Frame service when configured to support the transfer and radiation of AOS frames. These functions correspond to the operational scenario described in 2.5.3.

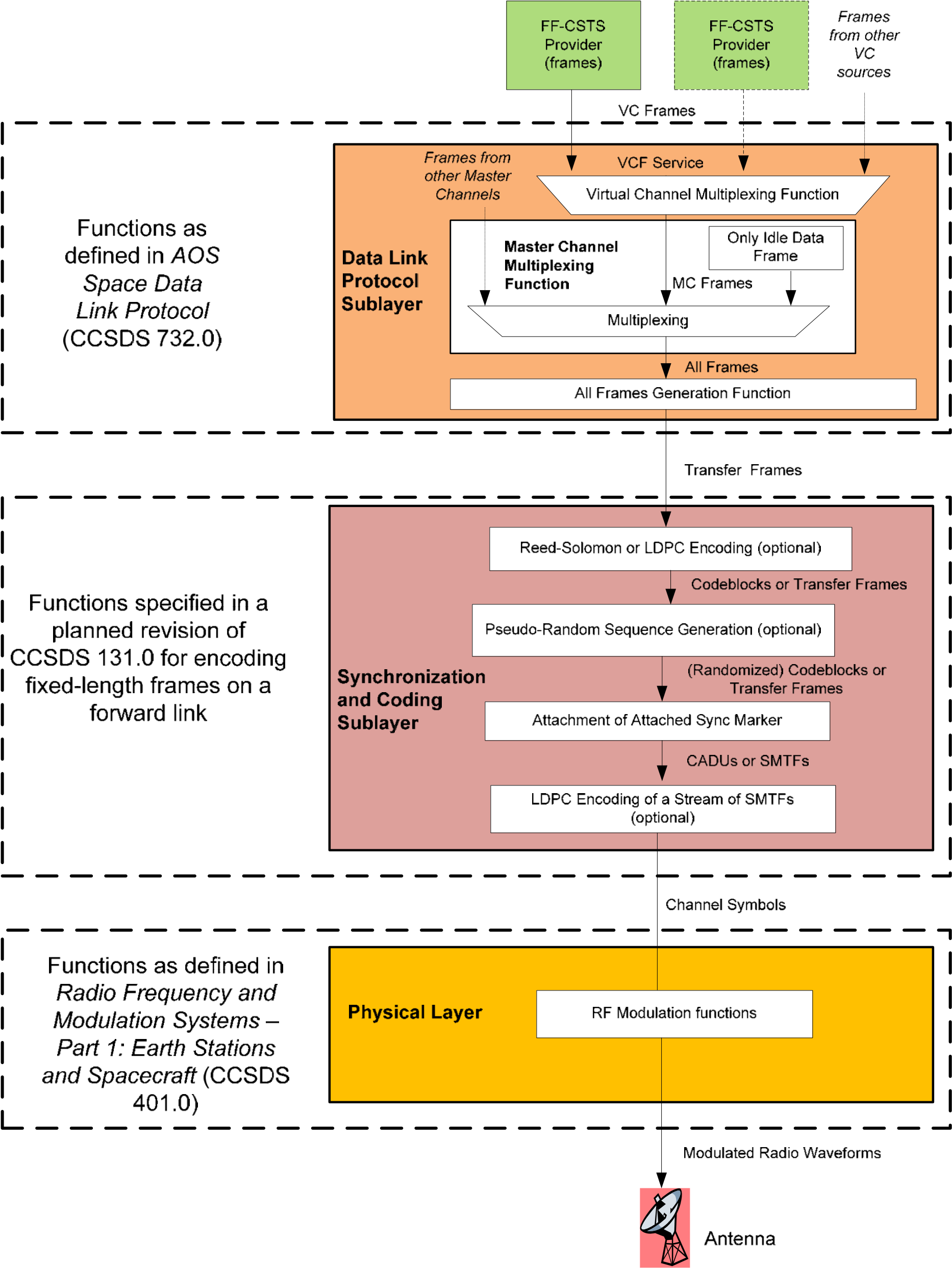
### I3.2 FORWARD FRAME CSTS PROVIDER

The FF-CSTS Provider represents an instance of an entity that performs the functions that are specified in this Recommended Standard. In the Transfer Frame configuration, multiple instances of the FF-CSTS Provider may exist for one space link physical channel. Each FFCSTS Provider supplies the frames from one virtual channel. In the AOS Frames configuration, the FF-CSTS Provider bit mask is configured to match data units for which the TFVN, SCID, and VCID fields in the first four octets of the AOS Transfer Frame Primary Header correspond to the GVCID of the virtual channel assigned to that transfer service instance.

### I3.3 DATA LINK PROTOCOL SUBLAYER

#### I3.3.1 General

For the AOS Frames configuration, the Data Link Protocol Sublayer functions are defined in the *AOS Space Data Link Protocol* Recommended Standard (reference [L13]). The functions of this sublayer that are relevant to the transfer of AOS Frames are the Virtual Channel Multiplexing Function, the Master Channel Multiplexing Functions, and the All Frames Generation Function.



#### Figure I-2: Forward Frame Service—Transfer Frame Configuration—AOS Frames I3.3.2 Virtual Channel Multiplexing Function

The Virtual Channel Multiplexing function of reference [L13] multiplexes the Transfer Frames from one or more VCs with the same MCID into a single stream of Transfer Frames. The sources of these VCs can be multiple FF-CSTS service instances, VC frames generated by the AOS SDLP’s Virtual Channel Generation Function (see reference [L13]) and/or other VC-generating sources of externally-generated transfer frames input via the AOS SDLP’s VC Frame Service (see reference [L13]).

The specification of the Virtual n Multiplexing function in reference [L13] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component VCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. Section G2.2.1.2 specifies three multiplexing schemes (absolute priority, polling vector, and FIFO) that must be implemented by any instance of the Virtual Channel Multiplexing function that receives its frames from an instance of the Forward Frame service.

NOTE – The VC Multiplexing function defined in reference [L13] also provides for generation of Only Idle Data frames in the absence of user-data-bearing Transfer Frames from any of the input VCs and when there is only a single MC on the Physical Channel. However, for the purposes of this example configuration, it is assumed that the Only Idle Data frames are generated by the Master Channel Multiplexing function (see I3.3.3).

#### I3.3.3 Master Channel Multiplexing Function

The Master Channel Multiplexing function of reference [L13] multiplexes the Transfer Frames from one or more Master Channels into a single stream of Transfer Frames destined for the space Physical Channel. The sources of these Master Channels can be one or more Virtual Channel Multiplexing function instances and/or other VC-generating sources of externally-generated transfer frames input via the AOS SDLP’s MC Frame Service (see reference [L13]).

The Master Channel Multiplexing function also generates Only Idle Data frames ‘to preserve the continuity of the transmitted stream in the event that there are no valid Transfer Frames available for transmission at a release time’ (4.2.6.4 of reference [L13]).

The specification of the Master Channel Multiplexing function in reference [L13] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component MCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. Section G2.2.2 specifies three multiplexing schemes (absolute priority, polling, and FIFO) that must be implemented by any instance of the Master Channel Multiplexing function that receives frames from any VC Multiplexing function instance used by a Forward Frame service instance.

#### I3.3.4 All Frames Generation Function

The All Frames Generation function of reference [L13] optionally adds an FECF to the trailer of each frame if the function is configured to do so.

NOTE – As specified in the AOS Space Data Link Protocol specification (reference [L13]), the All Frames Generation function is also ‘used to insert Insert Service data units into the Transfer Frames of a Physical Channel’. However, as currently defined the AOS protocol prohibits the concurrent use of the Insert Service with entities (such as the Forward Frame service) that provide frames via the VC Frames service interface of the Virtual Channel Multiplexing function. It is possible that a future version of the AOS SDLP may remove the constraints that prohibit concurrent use of the Insert Service and Forward Frame service, but at this time the Insert Service is not addressed by this Recommended Standard.

### I3.4 SYNCHRONIZATION AND CHANNEL CODING SUBLAYER

NOTE – As of the publication of this Recommended Standard, the Synchronization and Channel Coding sublayer functions used to support fixed-length frames such as AOS and fixed-length USLP frames on the forward link are planned to be specified in a revised version of reference [L15], which will recommend the use of Reed-Solomon and LDPC encoding for forward link frames, and LDPC coding of a stream of SMTFs (see 1.6.1.6 d)).

The functions of reference [L15] that are relevant to the Forward Frame service are:

1. the choice between Reed-Solomon, LDPC encoding of individual frames, LDPC coding of a stream of SMTFs, or no encoding of individual transfer frames;
2. the (optional) Pseudo-Randomizer function; and
3. attachment of Attached Sync Marker.

Reference [L15] specifies the permissible combinations of these synchronization and channel coding functions.

### I4 TRANSFER FRAME CONFIGURATION FOR VARIABLE-LENGTH USLP FRAMES I4.1 GENERAL

Figure I-3 depicts the functions that comprise the provision and production of the Forward Frame service when configured to support the transfer and radiation of variable-length USLP frames.

### I4.2 FORWARD FRAME CSTS PROVIDER

The FF-CSTS Provider represents an instance of an entity that performs the functions that are specified in this Recommended Standard. In the Transfer Frame configuration, multiple instances of the FF-CSTS Provider may exist for one space link physical channel. Each FFCSTS Provider supplies the frames from one virtual channel. In the variable-length USLP Frames configuration, the FF-CSTS Provider bit mask is configured to match data units for which the TFVN, SCID, and VCID fields in the first four octets of the USLP Transfer Frame Primary Header correspond to the GVCID of the virtual channel assigned to that transfer service instance.

### I4.3 DATA LINK PROTOCOL SUBLAYER

#### I4.3.1 General

For the variable-length USLP Frames configuration, the Data Link Protocol Sublayer functions are defined in the *Unified Space Data Link Protocol* Recommended Standard (reference [L14]). The functions of this sublayer that are relevant to the transfer of USLP Frames are the Virtual Channel Multiplexing Function, the Master Channel Multiplexing Functions, and the All Frames Generation Function.

#### I4.3.2 Virtual Channel Multiplexing Function

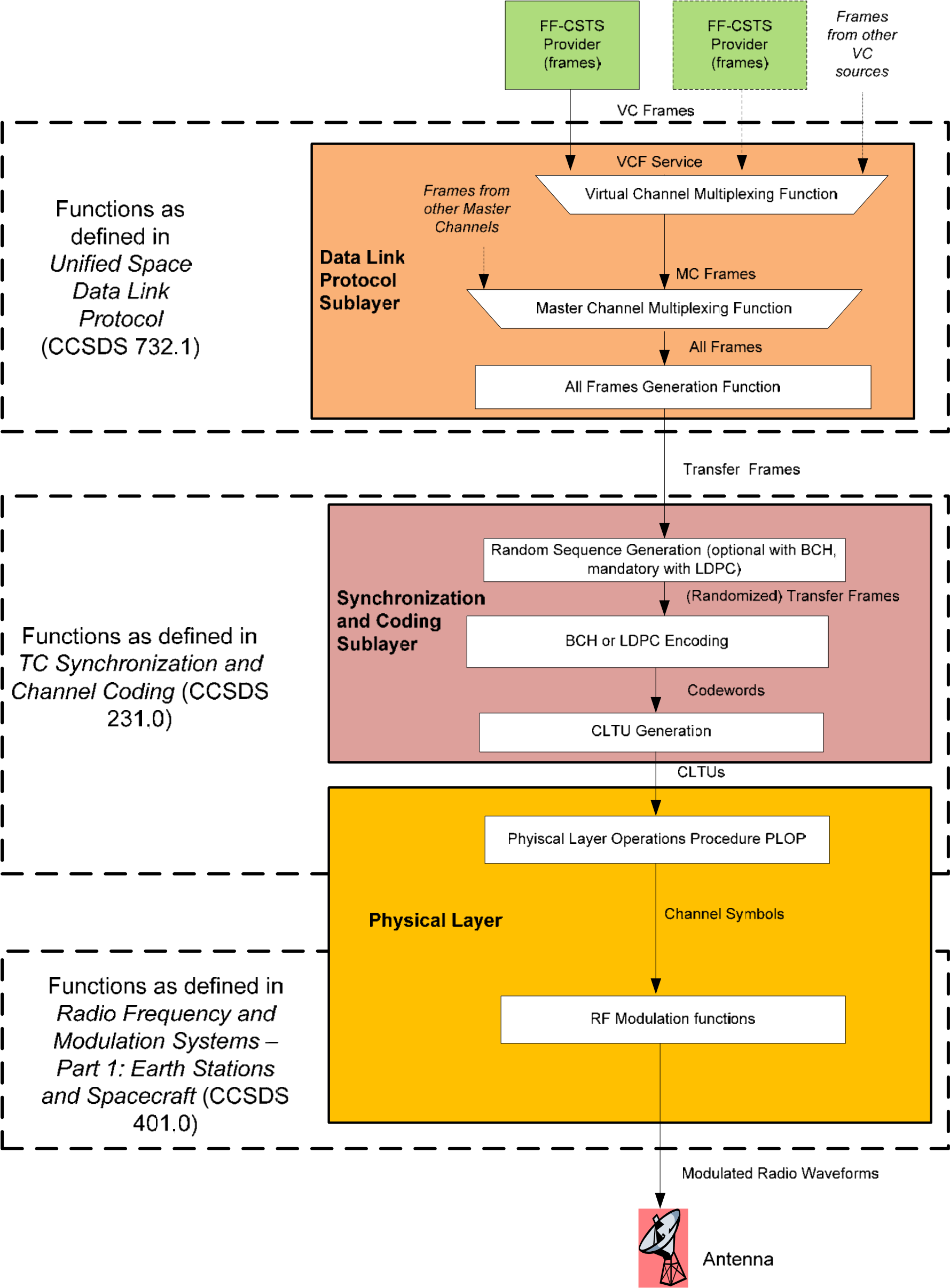
The Virtual Channel Multiplexing function of reference [L12] multiplexes the Transfer Frames from one or more VCs with the same MCID into a single stream of Transfer Frames. The sources of these VCs can be multiple FF-CSTS service instances, VC frames generated by the USLP’s Virtual Channel Generation Function (see reference [L14]) and/or other VCgenerating sources of externally-generated transfer frames input via the USLP’s VC Frame Service (see reference [L14]).

The specification of the Virtual Channel Multiplexing function in reference [L14] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component VCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. Section G2.2.1 specifies three multiplexing schemes (absolute priority, polling, and FIFO) that must be implemented by any instance of the Virtual Channel Multiplexing function that receives its frames from an instance of the Forward Frame service.

#### I4.3.3 Master Channel Multiplexing Function

The Master Channel Multiplexing function of reference [L14] multiplexes the Transfer Frames from one or more Master Channels into a single stream of Transfer Frames destined for the space Physical Channel. The sources of these Master Channels can be one or more Virtual Channel Multiplexing function instances and/or other VC-generating sources of externallygenerated transfer frames input via the USLP’s MC Frame Service (see reference [L14]).

The specification of the Master Channel Multiplexing function in reference [L14] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component MCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. Annex subsection G2.2.2 specifies three multiplexing schemes (absolute priority, polling, and FIFO) that must be implemented by any instance of the Master Channel Multiplexing function that receives frames from any VC Multiplexing function instance used by a Forward Frame service instance.



**Figure I-3: Forward Frame Service—Transfer Frame Configuration—VariableLength USLP Frames**

#### I4.3.4 All Frames Generation Function

The All Frames Generation function of reference [L14] optionally adds an FECF to the trailer of each frame. For USLP, only single Transfer Frames are submitted to the TC Synchronization and Channel Coding sublayer (see reference [L11]) to be transformed into a single CLTU.

When being submitted to the underlying Synchronization and Channel Coding sublayer (I4.4), the same frame may be submitted multiple times. The number of times a frame is submitted is based on configuration parameters set by Service Management that specify how many times Sequence-Controlled (type AD) and COP Control Command (type BC) USLP frames for each virtual channel are to be repeated (Expedited [type BD] frames are not repeated) (see reference [L14]).

NOTE – As specified in references [L11] and [L14], the Data Link Protocol Sublayer functions include the capability to cause the Synchronization and Channel Coding Sublayer (I2.4) to repeat AD and BC frames within CLTUs. The mechanism for accomplishing this is formally specified in references [L11] as a ‘data unit’ consisting of a group of frames (for USLP this group is limited to a single frame) to be transmitted in a single CLTU, accompanied by a *Repetitions* parameter that specifies the number of times that the resultant CLTU is to be transmitted. The same effect on the space link can be accomplished by having the Data Link Protocol Sublayer submit the frame to the Synchronization and Channel Coding Sublayer *Repetitions* times. Having the frame submitted to the Sync and Channel Coding Sublayer *Repetitions* times is the way that the repetition capability is implemented for SLE Forward CLTU (reference [L4]) and Forward Space Packet (FSP - reference [L19]): for the F-CLTU service, the service user transfers the same (already formed) CLTU *Repetitions* times, and for the FSP service, the Data Link Protocol Sublayer forms and submits the groupof-frames data unit to the Synchronization and Channel Coding Sublayer *Repetitions* times (based on Service Management configuration parameters). For consistency with those SLE implementations and in recognition that Forward Frame service instances may share space links with FSP service instances, the description of the repetition capability in this informative annex is phrased in terms of the approach in which the Data Link Protocol Sublayer submits the USLP frame to the Synchronization and Channel Coding Sublayer *Repetitions* times.

### I4.4 SYNCHRONIZATION AND CHANNEL CODING SUBLAYER

For the variable-length USLP frames configuration, the Synchronization and Channel Coding sublayer functions are the same as those identified in I2.4.

### I5 TRANSFER FRAME CONFIGURATION FOR FIXED-LENGTH USLP FRAMES I5.1 GENERAL

Figure I-4 depicts the functions that comprise the provision and production of the Forward Frame service when configured to support the transfer and radiation of fixed-length frames.

### I5.2 FORWARD FRAME CSTS PROVIDER

The FF-CSTS Provider represents an instance of an entity that performs the functions that are specified in this Recommended Standard. In the Transfer Frame configuration, multiple instances of the FF-CSTS Provider may exist for one space link physical channel. Each FF-CSTS Provider supplies the frames from one virtual channel. In the fixed-length USLP Frames configuration, the FF-CSTS Provider bit mask is configured to match data units for which the TFVN, SCID, and VCID fields in the first four octets of the USLP Transfer Frame Primary Header correspond to the GVCID of the virtual channel assigned to that transfer service instance.

### I5.3 DATA LINK PROTOCOL SUBLAYER

#### I5.3.1 General

For the fixed-length USLP Frames configuration, the Data Link Protocol Sublayer functions are defined in the *Unified Space Data Link Protocol* Recommended Standard

(reference [L14]). The functions of this sublayer that are relevant to the transfer of USLP Frames are the Virtual Channel Multiplexing Function, the Master Channel Multiplexing Functions, and the All Frames Generation Function.

#### I5.3.2 Virtual Channel Multiplexing Function

The Virtual Channel Multiplexing function of reference [L14] multiplexes the Transfer Frames from one or more VCs with the same MCID into a single stream of Transfer Frames. The sources of these VCs can be multiple FF-CSTS service instances, VC frames generated by the USLP’s Virtual Channel Generation Function (see reference [L14]) and/or other VC-generating sources of externally-generated transfer frames input via the USLP’s VC Frame Service (see reference [L14]).

The specification of the Virtual Channel Multiplexing function in reference [L14] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component VCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. Section G2.2.1 specifies three multiplexing schemes (absolute priority, polling, and FIFO) that must be implemented by any instance of the Virtual Channel Multiplexing function that receives its frames from an instance of the Forward Frame service.

NOTE – The VC Multiplexing function defined in reference [L14] also provides for generation of Only Idle Data frames in the absence of user-data-bearing Transfer

Frames from any of the input VCs and when there is only a single MC on the Physical Channel. However, for the purposes of this example configuration, it is assumed that the Only Idle Data frames are generated by the Master Channel Multiplexing function (see I5.3.3).

#### I5.3.3 Master Channel Multiplexing Function

The Master Channel Multiplexing function of reference [L14] multiplexes the Transfer Frames from one or more Master Channels into a single stream of Transfer Frames destined for the space Physical Channel. The sources of these Master Channels can be one or more Virtual Channel Multiplexing function instances and/or other VC-generating sources of externallygenerated transfer frames input via the USLP’s MC Frame Service (see reference [L14]).

The Master Channel Multiplexing function also generates Only Idle Data frames ‘to preserve the continuity of the transmitted stream in the event that there are no valid Transfer Frames available for transmission at a release time’ (4.2.8.4 of reference [L14]).

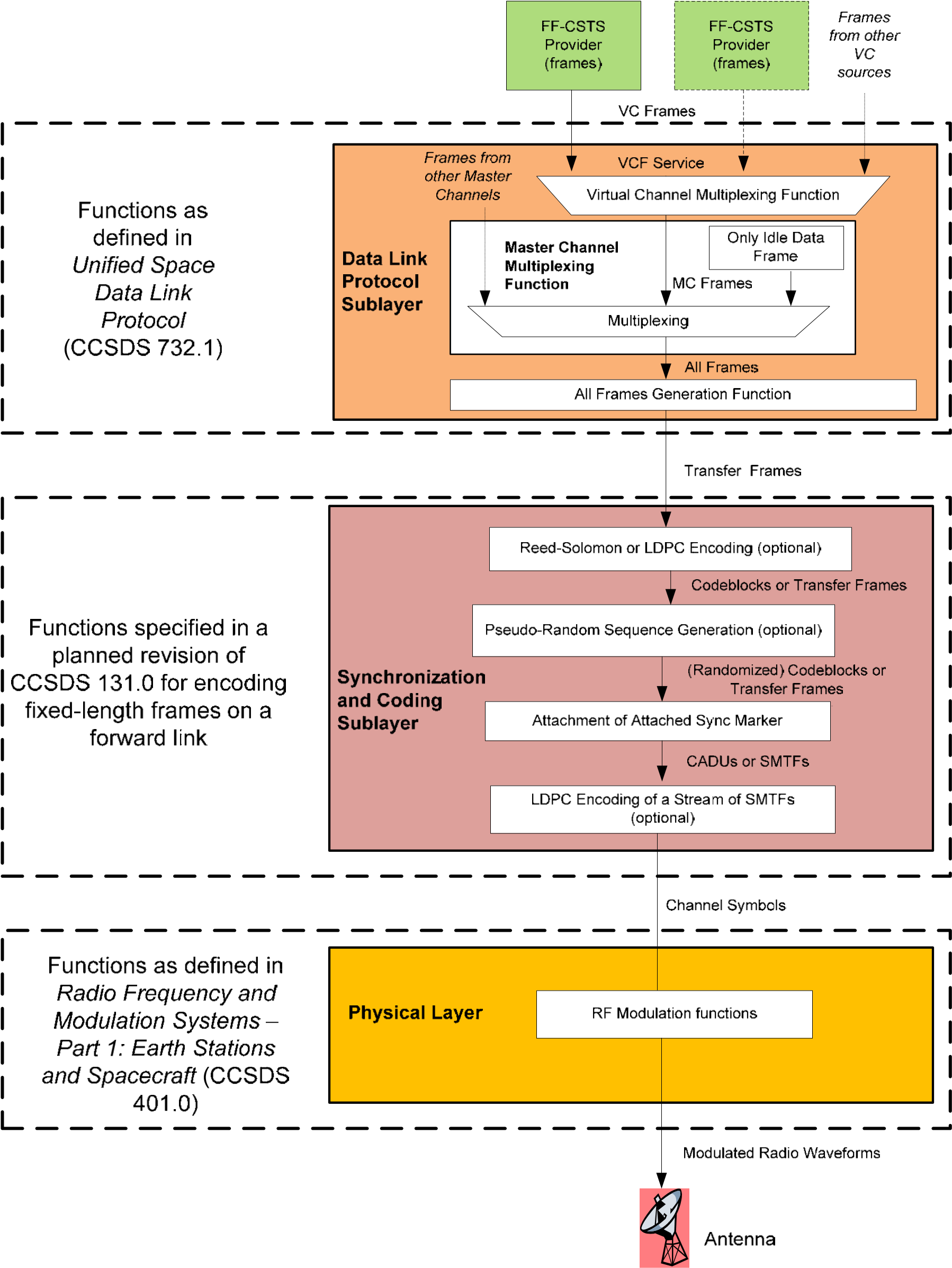
The specification of the Master Channel Multiplexing function in reference [L14] allows for the implementation of a multiplexing scheme that determines the order in which frames from the component MCs are multiplexed into the resulting master channel, but defers specification of the details of such a multiplexing scheme. Annex subsection G2.2.2 specifies three multiplexing schemes (absolute priority, polling, and FIFO) that must be implemented by any instance of the Master Channel Multiplexing function that receives frames from any VC Multiplexing function instance used by a Forward Frame service instance.

#### I5.3.4 All Frames Generation Function

The All Frames Generation function of reference [L14] optionally adds an FECF to the trailer of each frame if the function is configured to do so.

NOTE – As specified in the Unified Space Data Link Protocol specification (reference [L14]), the All Frames Generation function is also ‘used to insert Insert Service data units into the Transfer Frames of a Physical Channel’. However, as currently defined USLP puts constraints on entities (such as the Forward Frame service) that provide frames via the VC Frames service interface of the Virtual Channel Multiplexing function. These constraints make the concurrent use of the Insert Service with Forward Frame service impractical. It is possible that a future version of USLP may remove the constraints that prohibit concurrent use of the Insert Service and Forward Frame service, but at this time the Insert Service is not addressed by this Recommended Standard.

The insertion of Insert Service data units is optional, and its use is outside the scope of the Forward Frame service. However, if the Insert Service is used on a given Physical Channel, the users of all Forward Frame service instances associated with that Physical Channel must generate their transfer frames with empty Insert Zones corresponding to the size of the Insert Service data units. The presence or absence of the Insert Zones, and their size if present, are coordinated through Service Management.



**Figure I-4: Forward Frame Service—Transfer Frame Configuration—Fixed-Length USLP Frames**

### I5.4 SYNCHRONIZATION AND CHANNEL CODING SUBLAYER

For the fixed-length USLP frames configuration, the Synchronization and Channel Coding sublayer functions are the same as those identified in I3.4.

### I6 CADU CONFIGURATION I6.1 GENERAL

The CADU configuration is used when the Forward Frame service user generates and transfers to the ESLT CADUs, which are fixed-length data units that are continuously and contiguously into the forward link. The CADU configuration supports the transfer and transmission of CADUs bearing AOS or USLP fixed-length frames.

The processing functions performed by the ESLT in the CADU configuration are:

1. Multiplexing of Only Idle Data CADUs into the stream of user-supplied CADUs to preserve the continuity of the transmitted stream in the event that there are no valid CADUs available for transmission at a release time;
2. RF modulation; and
3. Radiation of the signal from the antenna.

The nominal use case for the CADU configuration is it’s use by rudimentary ESLTs that do not perform CCSDS space link protocol functions or any CCSDS frame coding, randomization, or synchronization marker attachment functions. However, the CADU configuration can also be used by missions that use mission-specific frame protocols and/or coding schemes for their frames, which result in fixed-length data units that must be transmitted continuously and contiguously.

The CADU configuration does not accommodate the AOS or USLP Insert Service because any encoding of the frames enclosed by the CADUs must include the data in the Insert Zones, but with the Insert Service this data is not put into Insert Zones until after the frames have been encoded.

Figure I-5 depicts the functions that comprise the provision and production of the Forward Frame service when configured to support the transfer and radiation of CADUs. These functions correspond to the operational scenario described in 2.5.4.

### I6.2 FORWARD FRAME CSTS PROVIDER

The FF-CSTS Provider represents an instance of an entity that performs the functions that are specified in this Recommended Standard. In the CADU configuration, only one instance of the FF-CSTS Provider may exist for one space link physical channel. In the CADU configuration, the FF-CSTS Provider bit mask is configured to match data units for which the first four octets of the CADU correspond to the first four octets of the Attached Sync Marker of the frame synchronization scheme used on the physical channel.

### I6.3 SYNCHRONIZATION AND CHANNEL CODING SUBLAYER

#### I6.3.1 General

For the CADU configuration, the Synchronization and Channel Coding Sublayer consists of the Only Idle Data CADU Generation and Multiplexing function.

#### I6.3.2 Only Idle Data CADU Generation and Multiplexing Function

The Only Idle Data CADU Generation and Multiplexing function (defined in G3.2.1) does not map directly to any function of a CCSDS Recommended Standard for the processing of fixed-length transfer frames. It is a function that must necessarily be performed in support of the Forward Frame service operating in the CADU configuration. In the CADU configuration, there is no multiplexing of virtual channels or master channels, and so the Only Idle Data Frame generation functions that are performed as part of those multiplexing functions in CCSDS fixed-frame SDLPs are not available for the stream of CADUs that bypass those SDLP multiplexing functions. The Only Idle Data CADU Generation and Multiplexing function preserves the continuity of the transmitted stream in the even that there are no valid CADUs available for transmission at release time. The resulting waveform is therefore conformant with that generated by a system that performs the appropriate CCSDS-standard SDLP (AOS or USLP for fixed-length frames) over the CCSDS-standard Synchronization and Channel Coding functions for fixed-length frames.

The Only Idle Data CADU Generation and Multiplexing function is categorized as a Synchronization and Channel Coding function because its purpose is to maintain the synchronization of the forward link in the absence of valid user-data-bearing CADUs.

**Synchronization**

**and Coding**

**Sublayer**

**Physical Layer**

CADUs

FF-CSTS

Provider

(

CADUs

)

CADU Multiplexing

Continuous and contiguous

stream of CADUs

Channel Symbols

Functions as defined in

*Radio Frequency and*

*Modulation Systems –*

*Part 1: Earth Stations*

*and Spacecraft*

(

CCSDS

401.0)

RF Modulation functions

Modulated Radio Waveforms

Antenna

Production functions as

defined in this

Recommended Standard

(

CCSDS

922.3)

Only Idle Data

CADU

#### Figure I-5: Forward Frame Service—CADU Configuration

**ANNEX J**

**EXAMPLE EXTENDED USE CASES FOR**

**THE FORWARD FRAME SERVICE**

**(INFORMATIVE)**

### J1 GENERAL

The Forward Frame service formally defined to support the transfer of TC Transfer Frames, AOS Transfer Frames, USLP Transfer Frames (both fixed and variable length), and CADUs. However, the Forward Frame service itself does not concern itself with the contents of the data units being transferred beyond two minimal criteria:

1. That the bits of the first four octets of the data unit that correspond to the subfields defined by the gvcid-bit-mask parameter, match the corresponding bits of the authorized-gvcid parameter.
2. That the data unit be no smaller than the size specified by the minimum-framelength parameter, and no larger than the size specified by the maximum-framelength parameter. The sizes permitted by these parameters range from one to 4294967295.

As defined for use with the CCSDS protocols listed above, the subfields in gvcid-bitmask correspond to the TFVN, SCID, and VCID subfields of the respective Transfer Frame Primary Headers. The Forward Frame service uses these parameters to ensure that only the frames for the VC that a Forward Frame service instance is authorized to transfer are processed by that service instance. But the subfields can be made to specify any subfields of those four octets, including none at all. In fact, for the CADU configuration, both the gvcid-bit-mask parameter and the authorized-gvcid parameter are set to all zeroes, which are the ‘don’t care’ values that bypass header checking altogether.

In addition to the flexibility provided by the Forward Frame service provision itself, the service places almost no requirements (other than those specified in annex G) on the underlying space data link protocols, synchronization and coding schemes, and physical channel. This allows the Forward Frame service to be potentially useful in space communication configurations beyond those for which the service was explicitly developed. For example, in addition to the use of TM Sync and Channel Coding to encode fixed length AOS and USLP frames on the forward link as described in I3 and I5, as of the publication of this Recommended Standard, CCSDS is considering the use of the Serially Concatenated Convolutional (Turbo) Code, as specified in reference [L16], on the forward link. If and when this revision occurs, the Forward Frame service is already capable of delivering AOS and fixed-length USLP transfer frames to the SCCC synchronization and coding layer.

Page J-1

The following subsections briefly identify a few examples of such space communication configurations. This set of example extended use cases is not exhaustive: other extended use cases can be formulated that combine various aspects of these examples. Of course, in each of these extended use case examples, the differences in the configurations of the underlying production processes would have to be supported by the capabilities of the ESLT and the ability of Service Management to address such configurations.

### J2 SUPPORT FOR TRANSFER OF A MASTER CHANNEL

In the Transfer Frame configuration, the definition of the Forward Frame service limits each service instance to a single Virtual Channel. However, there may be situations in which it is desirable to transfer a full Master Channel. Such possible uses cases could include large, multi-component, multi-Agency spacecraft for which each component has its own MC supplied by its own Control Center, or a relay satellite in which the satellite uses its own MC for its operations and supports one or more other MCs for the end-point Space User Nodes (e.g., rovers).

The Forward Frame service can be configured to support transfer of a full MC by configuring the gvcid-bit-mask and authorized-gvcid parameters to operate on only the

TFVN and SCID fields (i.e., set the VCID field bits of each to all zeroes).

Depending on the implementation of an ESLT capable of supporting this extended use case, the associated production processes would have to allow either a single VC Multiplexing function instance to receive multiple VCs for the same MC through a single service interface, or allow the data flow to bypass the VC Multiplexing functions and interface directly to a Master Channel Multiplexing function instance via the MC Frames service interface.

### J3 SUPPORT FOR TRANSFER OF SYSTEMATIC-ENCODED CCSDSCOMPLIANT TRANSFER FRAMES

Some ESLTs may support multiplexing of CCSDS-compliant transfer frames but not perform the frame level error correction coding. When the applicable error correction coding is systematic and therefore leaves the Transfer Frame Primary Header undisturbed, such as in the case of CCSDS-compliant Reed-Solomon and LDPC frame encoding, the frame encoding can be performed in the Earth User Node prior to transfer of the coded frame via the Forward Frame service. In this extended use case, the same gvcid-bit-mask and authorizedgvcid parameter settings for the fixed-length frame SDLP (e.g., AOS or USLP) can be used for both uncoded and already-encoded frames because the TFVN, SCID, and VCID subfields appear in the same positions of the first four octets of the data unit that is transferred.

ESLTs that do not perform frame-level error correction encoding may still optionally randomize the already-encoded frames and attach the ASMs to form the resulting stream of CADUs.

Page J-2

ESLTs that do perform the full set of fixed-length frame synchronization and channel coding functions (including, for this example, Reed-Solomon encoding) can accommodate user Missions that transfer already-encoded simply be configuring (via Service Management) the production processing the treat the frames as uncoded frames, thereby bypassing the frame encoding function of the ESLT.

### J4 SUPPORT FOR NON-CCSDS-COMPLIANT SPACE DATA LINK TRANSFER OF NON-CCSDS-COMPLIANT CADUS

Some Missions may use non-CCSDS-compliant SDLPs and/or non-CCSDS-compliant sync and coding schemes to produce fixed-length data units that must be continuously and contiguously transmitted on the space link. If these non-CCSDS-compliant SDLPs also define their own fixed-pattern Only Idle Data frames, then the Forward Frame service can be used to support such protocols in the CADU configuration.

This extended use case can be supported by any ESLT that supports the CADU configuration by configuring the Only Idle Data CADU parameter (see G3.2.1.1) of the Only Idle Data CADU Generation and Multiplexing function to contain the Only Idle Data frame of the SDLP with the corresponding synchronization and channel coding transformations.

### J5 TRANSFER OF SYNC-MARKED TRANSFER FRAMES

When a Mission uses LDPC coding of a stream of SMTFs (see reference [L15] and 1.6.1.6 d)), the CADU configuration cannot be used to transfer CADUs because there is no fixed-pattern Only Idle Data CADU that can be configured for the Only Idle Data CADU Generation and Multiplexing function.

However, LDPCcoding of a stream of SMTFs can be supported by the Forward Frame service by a hybrid of the fixed-frame synchronization and channel coding functions of the fixed-length frame Transfer Frame configurations (see I3 and I5) and the CADU configuration (see I6). Specifically, a single Forward Frame service instance is configured to supply SMTFs to the LDPC Coding of a Stream of SMTFs function, as illustrated in figure J-1. The Only Idle Data CADU configuration parameter is populated with an Only Idle Data SMTF; that is, an Only Idle Data Transfer Frame of the appropriate SDLP, with an ASM attached. As shown in the figure, the frame-level encoding functions, pseudo-random sequence generation, and ASM attachment functions are bypassed in this configuration.

Support for this extended configuration requires that the ESLT have the capabilities to perform this combination of functions and the ability of Service Management to support the configuration.

When used to support Missions that use CCSDS-compliant space data link protocols and LDPC encoding of SMTF streams, this extended configuration produces waveforms that are fully CCSDS-compliant. However, this extended configuration can also support Missions for which either or both the transfer frame format or the sync marker format are not CCSDS-compliant; that is, only the LDPC slicing function that is performed by the ESLT is CCSDS-compliant.

Page J-3

**Synchronization**

**and Coding**

**Sublayer**

**Physical Layer**

Channel Symbols

Functions as defined in

*Radio Frequency and*

*Modulation Systems –*

*Part 1: Earth Stations*

*and Spacecraft*

(

CCSDS

401.0)

RF Modulation functions

Modulated Radio Waveforms

Antenna

Pseudo-Random Sequence Generation (optional)

Attachment of Attached Sync Marker

Reed-Solomon or LDPC Encoding (optional)

LDPC Encoding of a Stream of SMTFs

Functions specified in a

planned revision of

CCSDS 131.0 for encoding

fixed-length frames on a

forward link

SMTFs

FF-CSTS

Provider

)

CADUs

(

CADU Multiplexing

Continuous and contiguous

stream of SMTFs

Only Idle Data

CADU

Production functions as

defined in this

Recommended Standard

922.3)

CCSDS

(

Idle SMTFs

#### Figure J-1: Forward Frame Service—Extended CADU Configuration—SMTFs

Page J-4

**ANNEX K**

## USING THE FORWARD FRAME SERVICE TO SHARE ACCESS TO A

**FORWARD LINK WITH DTN BUNDLES**

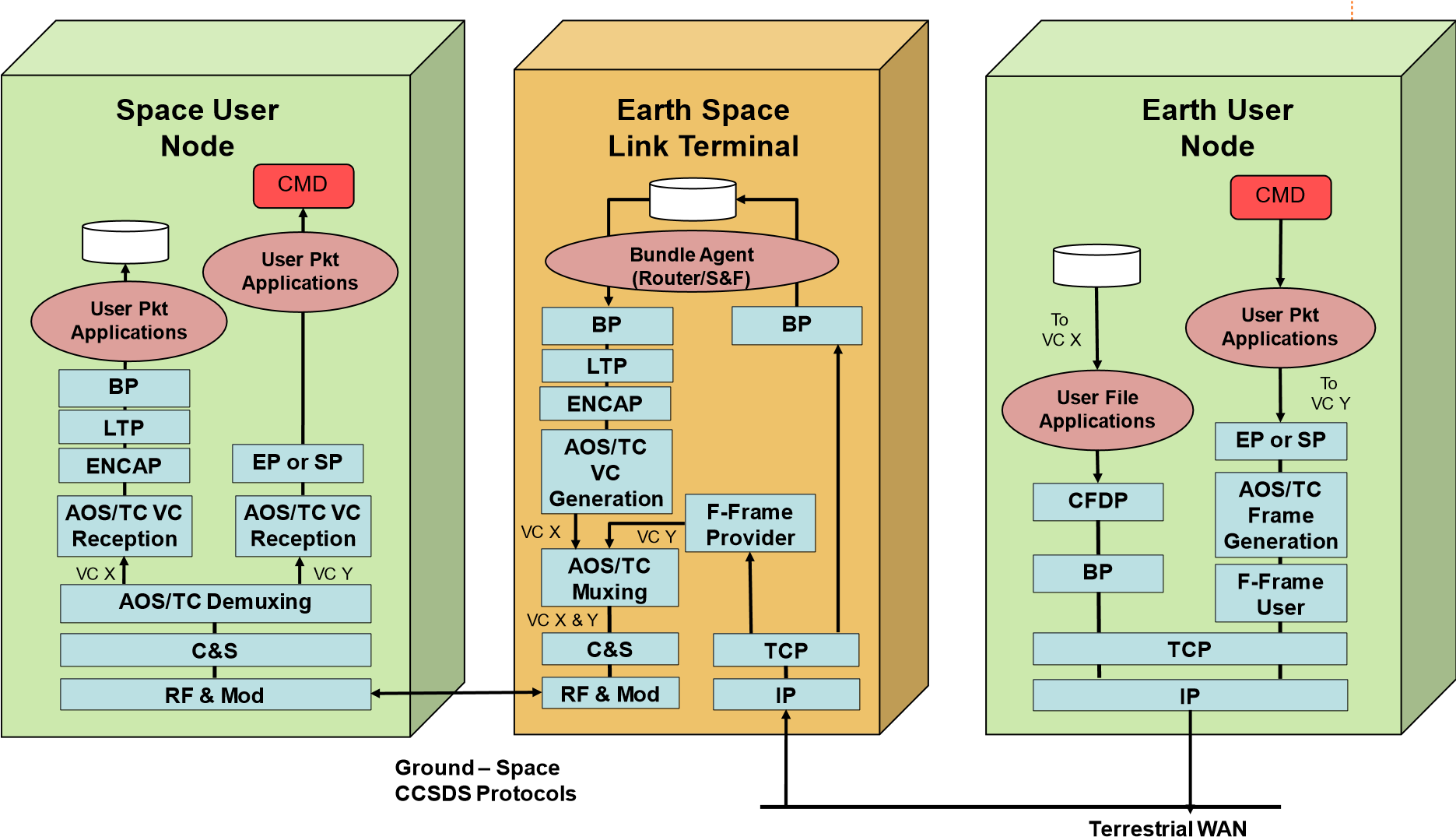
**(INFORMATIVE)**

As illustrated in annexes I and J, the Forward Frame service can be used to allow multiple sources of virtual channel frames to deliver those frames to an ESLT for multiplexing into a single forward link, as illustrated by the nominal and extended use case examples described in annex I and annex J. However, sharing of the same forward space link is not limited to only the use of multiple instances of the Forward Frame service. The ESLT may receive data destined for the shared space link in forms other than VC transfer frames, and the ESLT can embed that data into VCs that are then multiplexed with the VCs carried by Forward Frame service instances.

One such link-sharing scenario involves the case in which DTN (reference [L22]) *bundles* are transferred across the Internet to be multiplexed with Forward Frame-transferred transfer frames into a forward space link. This approach is recommended by the Interagency Operations Advisory Group (IOAG - reference [L25]) as a strategy for introducing DTN networking into communications with mission spacecraft, especially via an intermediate relay spacecraft. This use case assumes that relay missions that are service providers of DTN will want to separate use of DTN data for relay through the spacecraft, or for their own local DTN data destined for onboard payloads, but continue to use traditional CCSDS space link protocols for exchange of the relay spacecraft operational data (e.g., commands and telemetry). Virtual channel transfer frames are the lowest-level space data link protocol data units that can be transferred via the ESLT while still supporting multiplexing with DTN traffic. In this use case, transfer frames destined for the relay spacecraft operations functions would be delivered to the ESLT using FF-CSTS. DTN traffic destined for any DTN node, including any DTN nodes on board the relay spacecraft, would use the forward link shared access capability. This annex describes two examples of the protocol and service entities that would be performed by the ESLT when configured to support the DTN use case.

Figure K-1 illustrates the first example, and is an adaptation of figure 7-5 from the *Space Communications Cross Support - Architecture Description Document* (reference [L8]) that has been modified to show the multiplexing of the VCs in the ESLT in greater detail. This first example involves a single Earth User Node belonging to Agency ‘A’ communicating with a single Space User Node belonging to the same Agency through an ESLT operated by Agency ‘B’, in what is referred to as the ‘ABA’ configuration. The key feature of this example is that the Earth User Node simultaneously transfers DTN bundles and Earth User Node-generated VC transfer frames to the ESLT for transmission to the Space User Node over the same space data link. The particular example that is illustrated in the diagram shows the use of DTN to transfer files across the space link using CCSDS File Delivery Protocol (CFDP – reference [L23]) over the DTN bundle protocol (BP). From the perspective of the ESLT, the content of the DTN bundles is immaterial. Also, this example is cast in terms of using AOS or TC as the space data link protocol, but it is equally applicable when USLP is used. Additional details of this configuration can be found in reference [L8].

Page K-1



**Legend: AOS = Advanced Orbiting System BP = Bundle Protocol C&S = Coding & Synchronization ENCAP = Encapsulation EP = Encapsulation Packet LTP = Licklider Transmission Protocol RF & Mod = Radio Frequency & Modulation SP = Space Packet TC= Telecommand VC = Virtual Channel**

### Figure K-1: DTN ABA Architecture

Figure K-2 illustrates the second example addressed in this annex. The figure is an adaptation of figure 7-2 from the *Space Communications Cross Support - Architecture Requirements Document* (reference [L21]) that has been modified to show the multiplexing and demultiplexing of the VCs in greater detail. In this example, DTN protocols are used by an Earth User Node to send CFDP-segmented files through the ESLT to a Space Routing Node (e.g., a relay satellite) that relays the Earth-User Node-generated bundles to their ultimate destination, the Space User Node. The Space Routing Node in this example uses DTN for some file-related applications but uses traditional CCSDS space data link protocols for communication of spacecraft operations data. As in the first example, the Forward Frame CSTS instance in the ESLT provides the means for the Space Routing Node MOC to supply the Space Routing Node command-bearing transfer frames to be received by the ESLT for multiplexing into the space link with the DTN traffic destined for both the Space Routing Node and the Space User Node. Additional details of this configuration can be found in reference [L21].

Page K-2

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

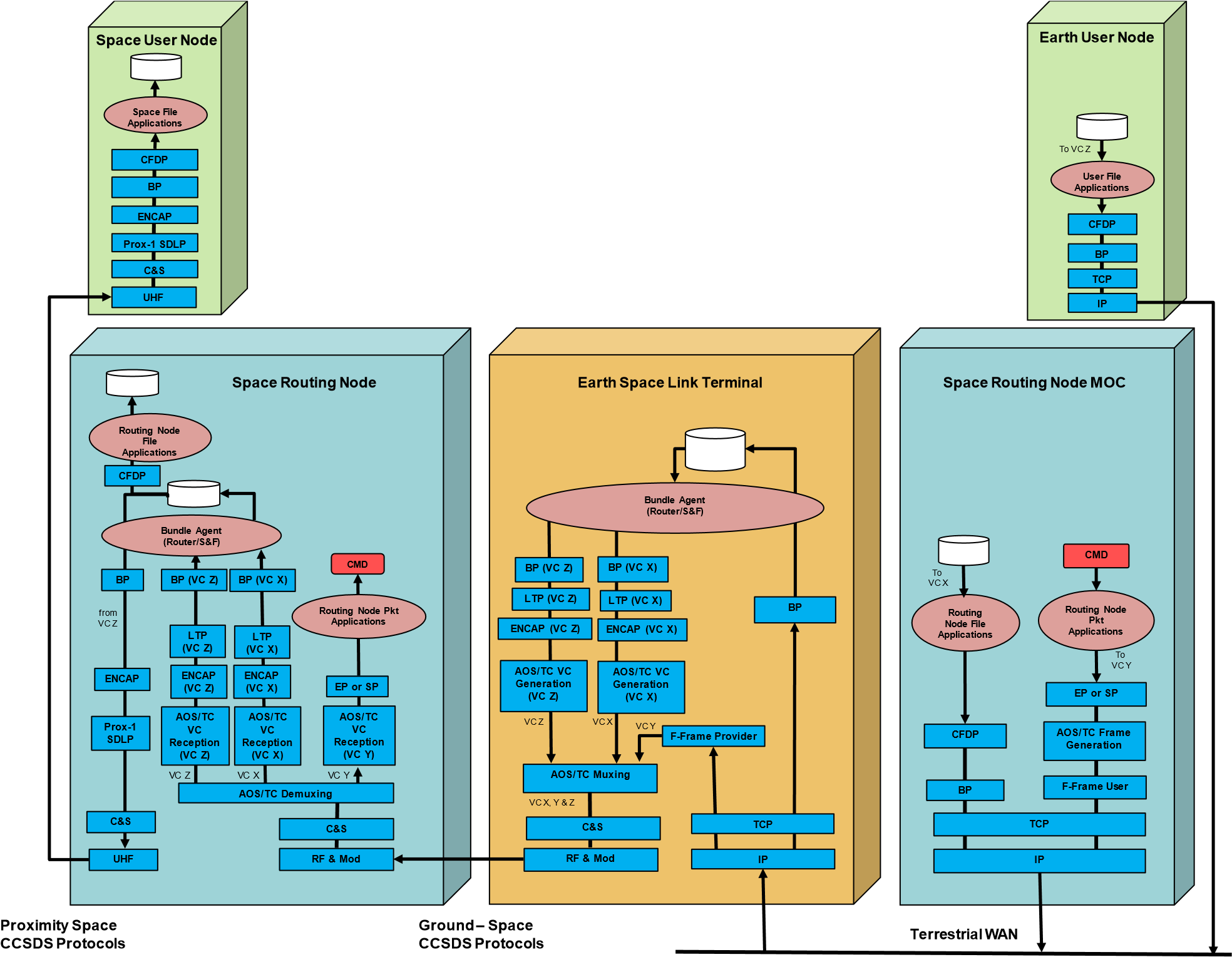
**CESG APPROVAL COPY - NOT FOR DISTRIBUTION**

CCSDS 922.3-B-0

Page K-3

December 2020

**CESG APPROVAL COPY - NOT FOR DISTRIBUTION**



**Legend: AOS = Advanced Orbiting System BP = Bundle Protocol C&S = Coding & Synchronization ENCAP = Encapsulation EP = Encapsulation Packet LTP = Licklider Transmission Protocol Prox-1 SDLP = Proximity 1 Space Data Link Protocol RF & Mod = Radio Frequency & Modulation SP = Space Packet**

**TC = Telecommand UHF = Ultra High Frequency VC = Virtual Channel**

### Figure K-2: DTN Relay Satellite Architecture

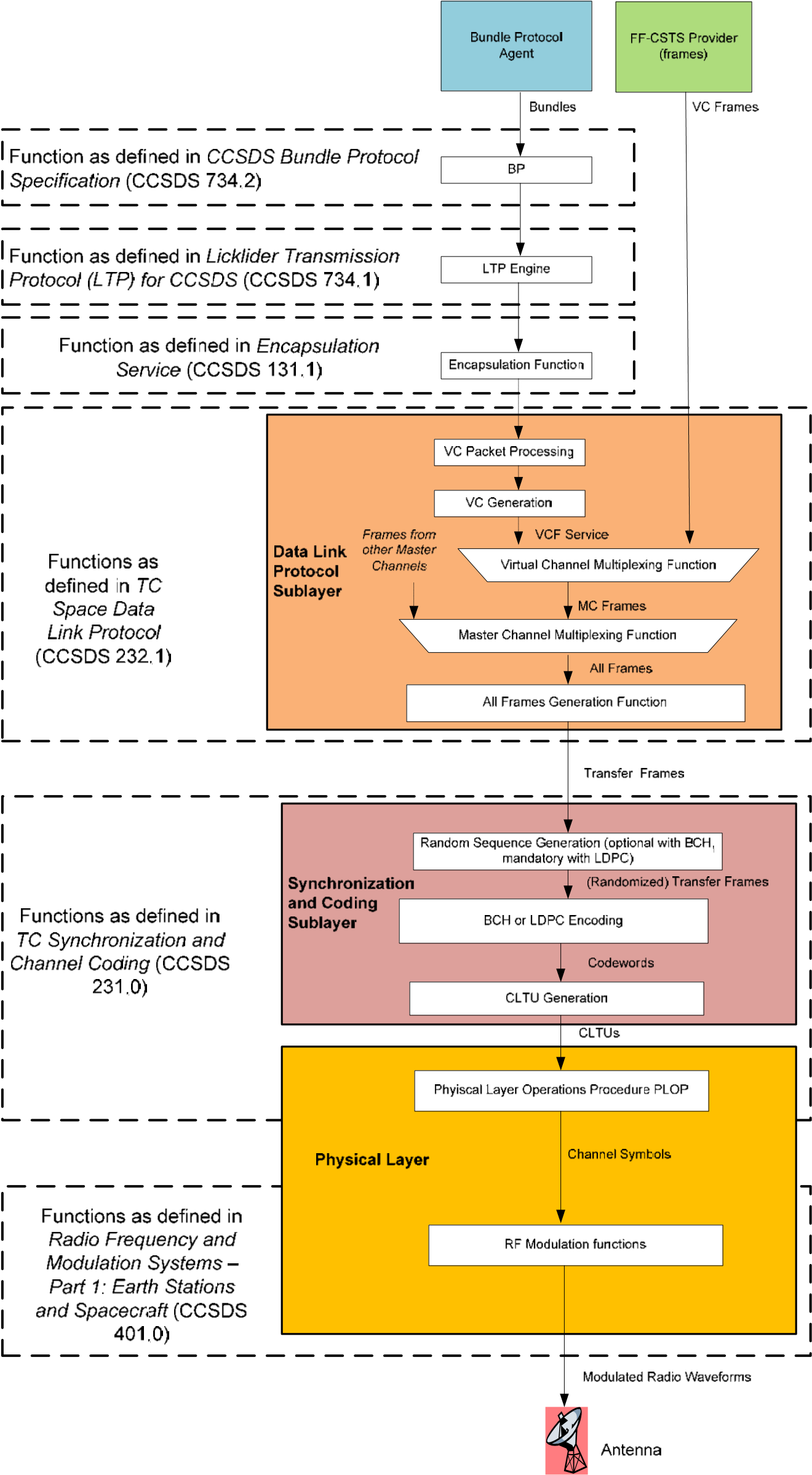
There are other possible DTN configurations, for example, one in which the space link between the Space Routing Node and the Space User Node uses a ‘long haul’ space data link protocol (TC, AOS, or USLP) instead of Prox-1. With respect to the Forward Frame service, the common characteristic is that the ESLT is able to combine DTN traffic and Usergenerated transfer frames into the single forward space link physical channel.

Figure K-3 shows the integration of DTN into the space link protocol stack for the transfer frame configuration for Telecommand frames that is described in I2. This protocol stack corresponds to the left-hand side protocol stack of the ESLT in figures K-1 and K-2, and identifies the specific CCSDS Recommended Standards that apply at each layer.

NOTE – Multiplexing of IP datagrams (instead of DTN bundles) into a shared forward link would be achieved by a comparable protocol stack that substitutes the Internet Protocol Extension function defined in the *IP over CCSDS Space Links* Recommended Standard (reference [L24]) for the Bundle Protocol Agent, BP, and LTP layers in figure K-3.

Page K-4

CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE



### Figure K-3: DTN and Forward Frame over Telecommand SDLP

CCSDS 922.3-B-0 Page K-5 December 2020 CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

**ANNEX L**

**INFORMATIVE REFERENCES**

**(INFORMATIVE)**

[L1] *Cross Support Concept—Part 1: Space Link Extension Services*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 910.3-G-3. Washington, D.C.: CCSDS, March 2006.

[L2] *Space Link Extension—Internet Protocol for Transfer Services*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 913.1-B-2. Washington, D.C.: CCSDS, September 2015.

[L3] *Cross Support Transfer Service Specification Framework Concept*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 920.0-G-1. Washington, D.C.: CCSDS, forthcoming.

[L4] *Space Link Extension—Forward CLTU Service Specification*. Issue 4. Recommendation for Space Data System Standards (Blue Book), CCSDS 912.1-B-4. Washington, D.C.: CCSDS, August 2016.

[L5] *Cross Support Service Management—Service Management Utilization Request Formats*. Recommendation for Space Data System Standards, under development.

[L6] *Cross Support Service Management—Simple Schedule Format Specification*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.1-B-1. Washington, D.C.: CCSDS, May 2018.

[L7] *Extensible Space Communication Cross Support—Service Management—Concept*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 902.0-G-1. Washington, D.C.: CCSDS, September 2014.

[L8] *Space Communications Cross Support—Architecture Description Document*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 901.0-G-1. Washington, D.C.: CCSDS, November 2013.

[L9] *Radio Frequency and Modulation Systems—Part 1: Earth Stations and Spacecraft*. Issue 30. Recommendations for Space Data System Standards (Blue Book), CCSDS 401.0-B-30. Washington, D.C.: CCSDS, February 2020.

[L10] *Data Transmission and PN Ranging for 2 GHz CDMA Link via Data Relay Satellite*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 415.1-B-1. Washington, D.C.: CCSDS, September 2011.

CCSDS 922.3-B-0 Page L-1 December 2020 CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

[L11] *TC Synchronization and Channel Coding*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 231.0-B-3. Washington, D.C.: CCSDS, September 2017.

[L12] *TC Space Data Link Protocol*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 232.0-B-3. Washington, D.C.: CCSDS, September 2015.

[L13] *AOS Space Data Link Protocol*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 732.0-B-3. Washington, D.C.: CCSDS, September 2015.

[L14] *Unified Space Data Link Protocol*. Issue 1. Draft Recommendation for Space Data System Standards (Blue Book), CCSDS 732.1-B-1. Washington, D.C.: CCSDS, October 2018.

[L15] *TM Synchronization and Channel Coding*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 131.0-B-3. Washington, D.C.: CCSDS, September 2017.

[L16] *Flexible Advanced Coding and Modulation Scheme for High Rate Telemetry Applications*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 131.2-B-1. Washington, D.C.: CCSDS, March 2012.

[L17] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. 2nd ed. International Standard, ISO/IEC 7498-1:1994. Geneva: ISO, 1994.

[L18] K. McCloghrie and M. Rose, eds. *Management Information Base for Network Management of TCP/IP-Based Internets: MIB-II*. STD 17 (RFC 1213). Reston, Virginia: ISOC, March 1991.

[L19] *Space Link Extension—Forward Space Packet Service Specification*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 912.3-B-3. Washington, D.C.: CCSDS, August 2016.

[L20] *Space Data Link Security Protocol*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 355.0-B-1. Washington, D.C.: CCSDS, September 2015.

[L21] *Space Communications Cross Support—Architecture Requirements Document*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 901.1-M-1. Washington, D.C.: CCSDS, May 2015.

[L22] *Rationale, Scenarios, and Requirements for DTN in Space.* Issue 1. Report Concerning Space Data Systems (Green Book), CCSDS 734.0-G-1. Washington, D.C.: CCSDS, August 2010.

CCSDS 922.3-B-0 Page L-2 December 2020 CCSDS RECOMMENDED STANDARD FOR CSTS FORWARD FRAME SERVICE

[L23] *CCSDS File Delivery Protocol (CFDP)*. Issue 5. Recommendation for Space Data System Standards (Blue Book), CCSDS 727.0-B-5. Washington, D.C.: CCSDS, July 2020.

[L24] *IP over CCSDS Space Links*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 702.1-B-1. Washington, D.C.: CCSDS, September 2012.

[L25] *Operations Concept for a Solar System Internetwork (SSI).* Report of the Interagency Operations Advisory Group Space Internetworking Strategy Group, IOAG.T.RC.001.V1. Washington, DC: IOAG, October 15, 2010.

CCSDS 922.3-B-0 Page L-3 December 2020

**ANNEX M**

**ABBREVIATIONS AND ACRONYMS**

## (INFORMATIVE)

|  |  |
| --- | --- |
|  |  |
| AOS | Advanced Orbiting System |
| ASM | attached synchronization marker |
| ASN.1 | Abstract Syntax Notation One |
| BCH | Bose–Chaudhuri–Hocquenghem |
| BFDP | buffered frame data processing [procedure] |
| CADU | channel access data unit |
| CCSDS | Consultative Committee for Space Data Systems |
| CFDP | CCSDS File Delivery Protocol |
| CLCW | communications link control word |
| CLTU | communication link transmission unit |
| COP | Communications Operation Procedure |
| CSSS | Cross Support Service System |
| CSTS | Cross Support Transfer Service |
| DTN | Delay-Tolerant Network |
| EM | element management |
| ESLT | earth space link terminal |
| FECF | frame error control field |
| FF | Forward Frame [CSTS] |
| FIFO | first-in-first-out |
| FR | functional resource |
| GVCID | global virtual channel identifier |
| ICS | implementation conformance statement |
| IOAG | Interagency Operations Advisory Group |
| IP | Internet Protocol |
| ISO | International Organization for Standardization |
| LDPC | low density parity check |
| MC | master channel |

Page M-1

|  |  |
| --- | --- |
| MCID | master channel identifier |
| MIB | management information base |
| MUE | mission user entity |
| OID | only idle data |
| PICS | protocol implementation conformance statement |
| PDU | protocol data unit |
| PLOP | physical layer operations procedure |
| PM | provision management |
| Prox-1 | Proximity-1 (space data link protocol) |
| RF | radio frequency |
| RL | requirements list |
| SANA | Space Assigned Numbers Authority |
| SC-CSTS | service control CSTS |
| SCFDP | sequence-controlled frame data processing [procedure] |
| SCID | spacecraft identifier |
| SDLP | space data link protocol |
| SDLS | space data link security |
| SFW | [CSTS] Specification Framework |
| SLE | Space Link Extension |
| SL-PDU | space link protocol data unit |
| SMTF | Sync-Marked Transfer Frame |
| SNMP | Simple Network Management Protocol |
| Sync | synchronization |
| TC | telecommand |
| TCP | transmission control protocol |
| TFVN | Transfer Frame Version Number |
| TM | telemetry |
| UM | utilization management |
| USLP | Unified Space Data Link Protocol |
| VC | virtual channel |
| VCID | virtual channel identifier |

Page M-2

**ANNEX N**

## CROSS REFERENCES TO CROSS SUPPORT TRANSFER SERVICE

**SPECIFICATION FRAMEWORK**

**(INFORMATIVE)**

Table N-1 lists the specific sections, subsections, paragraphs, and tables of the CSTS SFW (reference [1]) that are normatively referenced by this Recommended Standard, and identifies the sections, subsections, paragraphs, and tables of this Recommended Standard that make specific references to each of those sections/paragraphs in reference [1].

### Table N-1: Cross References to CSTS Specification Framework Subsections

|  |  |
| --- | --- |
| **Referenced CSTS SFW**  **Section/Subsection/**  **Paragraph/Table** | **Referencing Section/Subsection/Paragraph/Table of this Recommended Standard** |
| 1.6.3.1 | 1.5.2 f) |
| 2.2 | 1.5.2 f) |
| 2.2.2.2 | 8.3 |
| 3.2.1.2 | 8.2 |
| 3.2.3.6 | table 4-6, table 5-6 |
| 3.2.4 | H1.3.3 |
| 3.3.2.7 | C1 (4 references), D1 (2 references), E1 (2 references) |
| 3.3.2.7.1 | table 4-7, table 5-7 |
| 3.3.2.7.1 c) | 9.2 |
| 3.4.2.2.4.3 | 7.2.4 |
| 3.7.2.3.1 | table 4-7, table 5-7 |
| 3.11.2.2.3.2 a) | 8.4 |
| 3.11.2.2.3.2 b) | table 4-6, table 5-6, 8.5 |
| 3.13.1.1 | 6.3 |
| 4.2.2.3 | table 4-8, table 5-9 |
| 4.2.2.4 | table 4-6, table 5-6 |
| 4.2.2.5 | table 4-6, table 4-8, table 5-6, table 5-9 |
| 4.2.3 | table 4-6, table 5-6 |
| 4.3 | table 3-1, table 3-2, table A-5 |
| 4.3.5 | 7.2.1, 7.2.2, 7.2.3, 7.2.4 |
| 4.6 | table A-5 |
| 4.6.3.1 a) | table 4-7, table 5-7 |
| 4.6.3.2.3 | table 4-3, table 5-3 |
| 4.6.3.3.1 | table 5-6 |
| 4.6.3.3.6 | table 4-8, table 5-9 |
| 4.6.3.5.1.1 | table 4-8, table 5-9 |

Page N-1

|  |  |
| --- | --- |
| **Referenced CSTS SFW**  **Section/Subsection/**  **Paragraph/Table** | **Referencing Section/Subsection/Paragraph/Table of this Recommended Standard** |
| 4.6.3.5.2 | table 4-8, table 5-9 |
| 4.6.3.5.3 | table 4-8, table 5-9 |
| 4.6.3.6.2 a) | table 4-8, table 5-9 |
| 4.6.3.6.2 b) | table 4-8, table 5-9 |
| 4.6.3.7 | table 4-8, table 5-9 |
| 4.6.4.1.4 | 4.5.5 b), 5.5.5 b), table 4-7, table 5-7 |
| 4.6.4.2.3 | 5.6.7.2.1.1 a), A2.2 AV29 |
| 4.6.4.2.3 a) | table 4-8, table 5-9 |
| 4.6.4.2.3 b) | table 4-8, table 5-9 |
| 4.6.4.2.3 c) | table 4-6, table 4-8, table 5-9 |
| 4.6.7 | G2.2.1.3, G2.2.1.6, G2.2.2.3, G2.2.2.6, G2.3.2, G2.3.7 |
| 4.7 | table 3-2, table A-5 |
| 4.7.3.1 | 5.5.2 |
| 4.7.3.2 | 5.5.3, table 5-9 |
| 4.7.3.2.1.2 | table 5-3 |
| 4.7.3.2.1.4 | table 5-6 |
| 4.7.3.2.2.3 | table 5-7 |
| 4.7.3.2.2.4 | table 5-7 |
| 4.7.3.2.2.7 | table 5-3 |
| 4.7.3.2.2.11 | table 5-6, table 5-9 |
| 4.7.3.3 | 5.5.4 |
| 4.7.3.4 | 5.5.5 |
| 4.7.3.5 | 5.5.1 |
| 4.7.3.6 | 5.5.1 |
| 4.7.3.7 | 5.5.1 |
| 4.7.3.8 | 5.5.1 |
| 4.7.4.1 | 5.6.5.1.1 |
| 4.7.4.2 | 5.6.7.1 |
| 4.7.4.2.2 | A2.2 AV29 |
| 4.7.4.2.2.2 | table 5-5, table 5-6, table 5-9 |
| 4.7.5 | 7.4.1, 7.4.2, 7.4.3 |
| 4.8 | table 3-1, 4.1.1, 4.6.4.1.1, table A-5, C1 |
| 4.8.2.2 | 4.1.2 |
| 4.8.3.1 | 4.5.2 |
| 4.8.3.2 | 4.5.3 |
| 4.8.3.3 | 4.5.4 |
| 4.8.3.3.2 | table 4-6 |
| 4.8.3.3.2.5 | table 4-6, table 4-8 |
| 4.8.3.4 | 4.5.5 |
| 4.8.3.5 | 4.5.1 |
| 4.8.3.6 | 4.5.1 |

Page N-2

|  |  |
| --- | --- |
| **Referenced CSTS SFW**  **Section/Subsection/**  **Paragraph/Table** | **Referencing Section/Subsection/Paragraph/Table of this Recommended Standard** |
| 4.8.3.7 | 4.5.1 |
| 4.8.3.8 | 4.5.1 |
| 4.8.3.8 b) | table 4-8 |
| 4.8.3.8 c) | table 4-8 |
| 4.8.3.8 d) | table 4-8 |
| 4.8.3.8 e) | table 4-8 |
| 4.8.3.9 | 4.5.1 |
| 4.8.3.10 | 4.5.1 |
| 4.8.3.11 | 4.5.1 |
| 4.8.4.2.1 | 4.6.5.1.1 |
| 4.8.4.2.7.1 | 4.6.5.1.2.1 |
| 4.8.4.3.2 | table 4-6 |
| 4.8.4.3.3 | A2.2 AV29 |
| 4.8.4.3.3.1 | table 4-8 |
| 4.8.4.3.4 | A2.2 AV29 |
| 4.8.4.3.4.1 | table 4-5 |
| 4.8.5 | 7.3 |
| 4.9 | table 3-1, table 3-2, table A-5 |
| 4.9.5 | 7.7 |
| 4.10 | table 3-1, table 3-2, table A-5 |
| 4.10.5 | 7.5.3, 7.5.5 |
| 4.11 | table 3-1, table 3-2, table A-5 |
| 4.11.5 | 7.6.1, 7.6.3 |
| 4.12 | table 3-1, table 3-2, 6-3, table A-5 |
| 4.12.4.1.3.1 | 6.6.2.1 a), 6.8.2 |
| annex B | 2.5.2.2h) i) Note 2 |
| B2.5 | table 4-6, table 5-6 |
| F | A1.2 |
| F3.3 | 5.6.7.2.1.2, A2.2 (numerous instances in the text), table A-7, table A-8, A2.2 AV2, A2.2 AV9, table A-10, table A-11, table A-12, table A-13, A2.2 AV9, table A-14, A2.2 AV10, table A-15, table A-16, table A-17, table A-19, table A-20, A2.2 AV25, table A-21, table A-22, A2.2 AV27, table A-23, A2.2 AV29, table A-24, C1 (5 instances), D1 (3 instances), E1 (2 instances), F1. |
| F3.4 | A2.2 (numerous instances in the text), table A-6, table A-12, A2.2 AV9, A2.2 AV10, table A-15, table A-16, A2.2 AV12, A2.2 AV13, A2.2 AV14, table A-17, table A-18, A2.2 AV20, table A-19, A2.2 AV25, table A-21, table A-23, table A-24, C1 (2 instances), D1, E1 (2 instances) |
| F3.5 | A2.2 (3 instances in the text), table A-6, table A-7, table A-8, A2.2 AV2, table A-9, table A-10 |
| F3.8 | table A-17, table A-23 |

Page N-3

|  |  |
| --- | --- |
| **Referenced CSTS SFW**  **Section/Subsection/**  **Paragraph/Table** | **Referencing Section/Subsection/Paragraph/Table of this Recommended Standard** |
| F3.9 | table A-6, D1 (2 instances) |
| F3.10 | table A-17, table A-18, A2.2 AV19, A2.2 AV20, A2.2 AV21, table A-19, C1 (3 instances) |
| F3.12 | table A-19, A2.2, AV25, table A-24 |
| F3.13 | table A-19, A2.2 AV25 |
| F3.14 | A2.2, AV10, E1 (2 instances) |
| F3.16 | table 4-3, table 5-3 (4 instances), A2.2, C11 |
| annex L | 2.2.2.1 |
| G3 | 3.5 |
| H2.3 | H2.1 |
| table 4-71 | 6.8.1 |
| table 4-72 | 6.8.1 |
| table 4-73 | 6.8.2 |
| table 4-74 | 6.8.1 |

Page N-4

1. In terms of Service Provider behavior, handling of an incoming PROCESS-DATA invocation and handling of an incoming forward buffer containing only one PROCESS-DATA invocation are identical. [↑](#footnote-ref-1)