The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Practices

GUIDELINES FOR THE SPECIFICATION OF CROSS SUPPORT TRANSFER SERVICES

DRAFT RECOMMENDED PRACTICE

CCSDS 921.2-R-1

RED BOOK
November 2017

The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Practices

GUIDELINES FOR THE SPECIFICATION OF CROSS SUPPORT TRANSFER SERVICES

DRAFT RECOMMENDED PRACTICE

CCSDS 921.2-R-1

RED BOOK
November 2017

AUTHORITY

Issue:	Red Book, Issue 1
Date:	November 2017
Location:	Washington, DC, USA

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

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the e-mail address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
E-mail: secretariat@mailman.ccsds.org

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED PRACTICE IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Practice 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 e-mail address indicated on page i.

PREFACE

This document is a draft CCSDS Recommended Practice. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

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

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 921.2-R-1	Guidelines for the Specification of Cross Support Transfer Services, Draft Recommended Practice, Issue 1	November 2017	Current draft

CONTENTS

Section Page

1 INTRODUCTION 1-1

1.1 PURPOSE OF THIS RECOMMENDED PRACTICE 1-1
1.2 SCOPE 1-2
1.3 APPLICABILITY OF THIS RECOMMENDED PRACTICE 1-2
1.4 RATIONALE 1-2
1.5 DOCUMENT STRUCTURE 1-2
1.6 DEFINITIONS 1-5
1.7 REFERENCES 1-9

2 OVERVIEW 2-1

3 RULES FOR DEFINITION OF CROSS SUPPORT TRANSFER SERVICES 3-1

3.1 PROCEDURES OF THE CROSS SUPPORT TRANSFER SERVICES 3-1
3.2 CSTS STATE MACHINE 3-2
3.3 CROSS SUPPORT TRANSFER SERVICE IDENTIFIERS 3-2
3.4 ASSOCIATION CONTROL PROCEDURE 3-2
3.5 DIRECT ADOPTION OF PROCEDURES 3-3
3.6 DERIVATION OF PROCEDURES 3-4
3.7 DEFINITION OF SERVICE-ORIGINAL PROCEDURES 3-4
3.8 INSTANCES OF A PROCEDURE TYPE 3-5
3.9 OPTIONAL PROCEDURES 3-6
3.10 PROCEDURE IDENTIFIER FOR AN EXTENDED OR SERVICE-ORIGINAL PROCEDURE 3-6
3.11 DERIVATION OF OPERATIONS 3-6
3.12 DERIVING A NEW SERVICE FROM AN EXISTING CROSS SUPPORT TRANSFER SERVICE 3-15
3.13 BACKGROUND—CONFIGURATION OF CROSS SUPPORT TRANSFER SERVICES 3-16

4 STRUCTURE AND CONTENT OF A CSTS SPECIFICATION 4-1

4.1 OVERVIEW 4-1
4.2 FRONT COVER AND FRONT MATTER 4-1
4.3 ‘INTRODUCTION’ SECTION 4-1
4.4 ‘OVERVIEW OF THE <NAME OF SERVICE>’ SECTION 4-2
4.5 ‘COMPOSITION OF THE <NAME OF SERVICE>’ SECTION 4-5
4.6 REFINED, EXTENDED, AND SERVICE-ORIGINAL PROCEDURES SECTIONS 4-12

CONTENTS (continued)

Section — Page

4.7 'SETTING OF SERVICE MANAGEMENT AND CONFIGURATION PARAMETERS INHERITED FROM CSTS SPECIFICATION FRAMEWORK PROCEDURES' SECTION 4-24
4.8 <NAME OF SERVICE> SERVICE-SPECIFIC VERSIONS OF SERVICE-GENERIC PARAMETERS AND EVENTS 4-25
4.9 'REFINEMENT AND EXCLUSION OF DEFINITIONS OF CSTS SPECIFICATION FRAMEWORK PARAMETERS, EVENTS, DIRECTIVES, AND DIAGNOSTIC VALUES USED BY THE <NAME OF SERVICE>' SECTION 4-26
4.10 SERVICE OBJECT IDENTIFIERS MODULE ANNEX 4-29
4.11 PROCEDURE PDU SPECIFICATION ANNEXES 4-35
4.12 OBJECT IDENTIFIERS OF PROCEDURE PARAMETERS, EVENTS, AND DIRECTIVES 4-42
4.13 '<NAME OF SERVICE> PRODUCTION' ANNEX 4-43
4.14 IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA 4-44
4.15 SECURITY, SANA, AND PATENT CONSIDERATIONS 4-44
4.16 ACRONYMS 4-44
4.17 INFORMATIVE REFERENCES 4-44
4.18 IMPLEMENTATION OF A CROSS REFERENCE MATRIX TO THE SERVICE SPECIFICATION FRAMEWORK 4-45

ANNEX A SECURITY, SANA AND PATENT CONSIDERATIONS (INFORMATIVE) A-1
ANNEX B MAPPING OF EXTENSIONS TO CSTS SPECIFICATION FRAMEWORK ASN.1 DATA STRUCTURES (INFORMATIVE) B-1
ANNEX C EXAMPLE SPECIFICATION OF A CSTS WITH DERIVED OPERATION PARAMETERS (INFORMATIVE) C-1
ANNEX D ACRONYMS (INFORMATIVE) D-1
ANNEX E INFORMATIVE REFERENCES (INFORMATIVE) E-1
ANNEX F CROSS REFERENCES TO SPECIFIC (SUB)SECTIONS OF THE CSTS SPECIFICATION FRAMEWORK (INFORMATIVE) F-1

Figure

1-1 Cross Support Service Documentation 1-4
2-1 CSTS Framework Procedures and Operations 2-1
2-2 CSTS Directly Composed of CSTS Specification Framework Procedures 2-2
2-3 CSTS Composed of Derived and Standard CSTS Specification Framework Procedures 2-3
2-4 CSTS Composed of Service-Original and Standard CSTS Specification Framework Procedures 2-4

CONTENTS (continued)

Figure Page

2-5 CSTS Derived from an Abstract CSTS 2-5
2-6 CSTS Derivation Examples 2-6
4-1 Example Overview Section Part of the Table of Contents for a CSTS Specification 4-3
4-2 Example Object Identifiers Module 4-35

Table

4-1 Example of a Service Composition Table for a CSTS Comprising All Directly Adopted Procedures 4-6
4-2 Example of a Service Composition Table for a CSTS Containing a Derived Procedure 4-10
4-3 Example of a Service Composition Table for a CSTS Containing a Service-Original Procedure 4-11
4-4 Example of a Required Operations Table for a Hypothetical Temperatures Cyclic Report Derived Procedure 4-15
4-5 Example of a Required Operations Table for a Hypothetical Expedited Delivery Service-Original Procedure 4-16
4-6 START Extension Parameter 4-17
B-1 Extension Parameters for Adding Parameters to CSTS Specification Framework Procedure Operation Invocations, Returns, and Acknowledgements B-3
B-2 Extension Parameters for Adding `diagnostic` Values to CSTS Specification Framework Common Operation Negative Returns and Negative Acknowledgements B-6
B-3 Extension Parameters for Adding `diagnostic` Values to CSTS Specification Framework Procedure Operation Negative Returns B-7
B-4 Adding Notifiable Events to CSTS Specification Framework Procedure NOTIFY Invocations B-8
B-5 Procedure-Specific NOTIFY Invocation Extensions B-8
F-1 Cross Reference to CSTS Specification Framework Sections and Subsections F-1

1 INTRODUCTION

1.1 PURPOSE OF THIS RECOMMENDED PRACTICE

This Recommended Practice defines the guidelines for the specification of CCSDS-standard Cross Support Transfer Services (CSTSes) based to the possible extent on the CSTS Specification Framework Recommended Standard, reference [1]. Consequently the prime audience of this Recommended Practice is any CCSDS Working Group chartered to develop a new CSTS specification. However, in case an Agency or industry wish to develop a proprietary service taking advantage of the CSTS Specification Framework (reference [1]) and possibly readily available implementations of the procedures specified therein will need to follow the guidelines defined in this Recommended Practice as well. Implementers and users of CSTSes will find in this Recommended Practice background material helping them in understanding CSTS specifications beyond the level of detail provided in the related Concept report (reference [E5]).

Any CSTS specification should follow the this Recommended Practice to the greatest extent possible and practical, but if a service has circumstances that are not addressed by this document, that service specification can go beyond these guidelines as long as those extensions are explicitly identified in the service specification.

Using the terminology defined in reference [E6] ‘Cross Support Transfer Service’ is the common name for a category of Cross Support Services, each of which allows a cross support service element typically located at an Earth-space terminal that belongs to one Space Agency to provide a defined set of capabilities to service users located at Earth User Nodes that belong to other Space Agencies via a connection-oriented, access-controlled transfer of spaceflight mission-related data between those service providers and service users, realized through the invocation and performance of defined operations in accordance with defined procedures.

In order to facilitate the development of new CSTSes, CCSDS has developed a standard set of procedures and operations from which multiple CSTSes may be created with minimum redundant reinvention. These standard CSTS procedures and operations (defined in the CSTS Specification Framework Recommended Standard, reference [1]) are designed to be extended and/or refined in various combinations to suit the needs of specific CSTSes.

CSTSes may be concrete or abstract. A concrete service is intended to be implemented as specified; i.e., its specification is sufficiently detailed to allow the implementation of real systems that will interoperate. An abstract service specification is intended to serve as the basis for a family of services that all share the common procedures, operations, and parameters provided by that abstract service. An abstract service specification may lack some details necessary for the implementation of services in real systems. An example of such abstract service is shown in figure 2-6.

1.2 SCOPE

This Recommended Practice defines:

a) the rules by which a new CCSDS-standard CSTS can be created from the CSTS procedures and operations defined in the CSTS Specification Framework Recommended Standard (reference [1]) or from existing concrete or abstract CSTSes;

NOTE – Figure 2-6 shows an example for the derivation of concrete CSTSes from an abstract CSTS.

b) the technical information that is to be included in the specification of such a CSTS specification.

This Recommended Practice does not specify:

a) individual CSTSes;

b) the methods or technologies required to provide a suitable environment for communications;

c) the management activities necessary to schedule, configure, control, and/or monitor CSTSes; or

d) methods by which CSTSes can be derived from CSTS procedures and operations.

1.3 APPLICABILITY OF THIS RECOMMENDED PRACTICE

This Recommended Practice provides a basis for the development of specifications for CSTSes that are derived from the standard CCSDS procedures and operations defined for CSTSes in reference [1].

1.4 RATIONALE

The goal of this Recommended Practice is to support the creation of new CSTSes through the use of the standard CSTS procedures and operations.

1.5 DOCUMENT STRUCTURE

1.5.1 DOCUMENT ORGANIZATION

This document is organized as follows:

a) section 1 presents the purpose, scope, applicability and rationale of this Recommended Practice and lists the definitions, conventions, and references used throughout this Recommended Practice;

b) section 2 provides an overview of this Recommended Practice and its role in the creation of new Cross Support Transfer Services from the procedures and operations defined in the CSTS Specification Framework Recommended Standard, reference [1], as well as through derivation from existing CSTSes;

c) section 3 documents the rules for definition of Cross Support Transfer Services;

d) section 4 specifies the structure and content of a CSTS specification;

e) annex A covers the security, SANA and patent considerations relevant to the specification of CSTSes based on this Recommended Practice by pointing to the relevant material presented in annex H of reference [1];

f) annex B summarizes the extension points to the ASN.1 data structures specified in subsection F4 of the CSTS Specification Framework (reference [1]);

g) annex C provides an informative example of the key normative sections of a CSTS specification for a hypothetical service with derived operations;

h) annex D lists the acronyms that are used in this Recommended Practice;

i) annex E is a list of the informative references;

j) annex F is a list of references to specific sections/subsections of the CSTS Specifications Framework (reference [1]) as used in this Recommended Practice.

1.5.2 CROSS SUPPORT TRANSFER SERVICES DOCUMENTATION

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

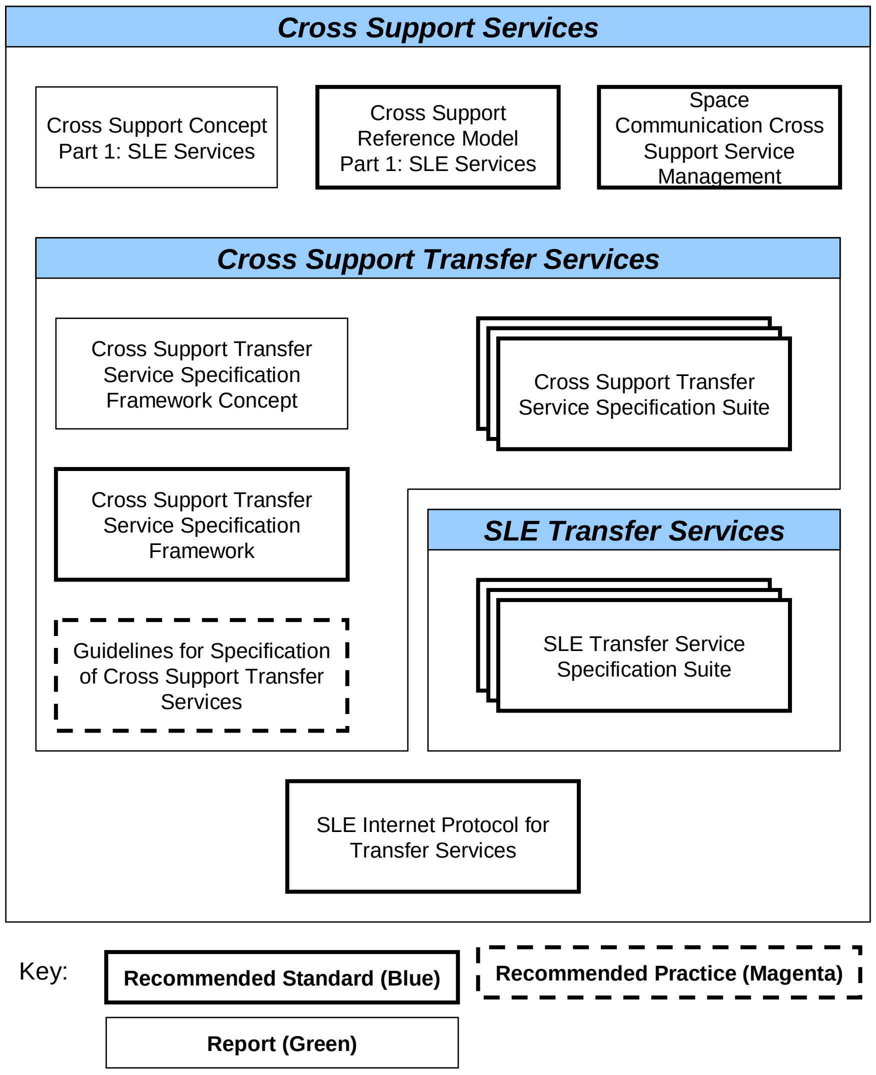
Cross Support Services
Cross Support Concept Part 1: SLE Services
Cross Support Reference Model Part 1: SLE Services
Space Communication Cross Support Service Management
Cross Support Transfer Services
Cross Support Transfer Service Specification Framework Concept
Cross Support Transfer Service Specification Suite
Cross Support Transfer Service Specification Framework
SLE Transfer Services
Guidelines for Specification of Cross Support Transfer Services
SLE Transfer Service Specification Suite
SLE Internet Protocol for Transfer Services
Key:
Recommended Standard (Blue)
Recommended Practice (Magenta)
Report (Green)

Figure 1-1: Cross Support Service Documentation

The Cross Support Service documents that are related to Cross Support Transfer Services are:

a) *Cross Support Concept—Part 1: Space Link Extension* (reference [E1]): 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 c) below);

b) *Cross Support Reference Model—Part 1: Space Link Extension Services* (reference [2]): a Recommended Standard that defines the CSTS Specification 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 1.6.1.1);

c) *Space Communication Cross Support Service Management* suite (references [E2] and [E3]): Future data format Recommended Standards will specify the Service Management Information Entities that are used to configure and schedule CSTSes;

d) *The SLE Transfer Services suite*: The SLE Transfer Services are a suite of cross support services that are used to transfer specific telecommand and telemetry protocol data units. 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 Specification Framework Recommended Standard (reference [1]). However, because of history (the SLE Transfer Services were already specified and implemented prior to development of the CSTS Specification Framework Recommended Standard (reference [1]) the SLE Transfer Services are separated from CSTSes;

e) *Space Link Extension—Internet Protocol for Transfer Services* (reference [E4])*:* A Recommended Standard that defines a protocol for transfer of Protocol Data Units (PDUs) defined in the Cross Support Transfer Services. The Recommended Standard was originally developed to support SLE transfer services (hence the title), but it is also applicable to (and specified for) use by Cross Support Transfer Services.

The documents specific to Cross Support Transfer Services are:

a) *Cross Support Transfer Services Specification Framework* (reference [1]): A Recommended Standard that defines the specification of the Cross Support Transfer Service (CSTS) procedures;

b) *Guidelines for the Specification of Cross Support Transfer Services* (this Recommended Practice): A Recommended Practice that defines the guidelines for construction of a Cross Support Transfer Service based on the CSTS Specification Framework;

c) *Cross Support Transfer Services Specification Framework Concept* (reference [E5]): A Report that provides tutorial material on the objectives and concepts of the CSTS Specification Framework;

d) *Cross Support Transfer Services Suite:* The set of specifications for actual CSTSes built from the procedures in the CSTS Specification Framework and in accordance with the 'Guidelines for the Specification of CSTSes' (this Recommended Practice).

1.6 DEFINITIONS

1.6.1 TERMS

1.6.1.1 Terms from the Cross Support Reference Model, Part 1: Space Link Extension

This Recommended Practice makes use of the following terms defined in reference [2]:

a) binding;

b) service agreement (called SLE Service Agreement);

c) service provider;

d) service user.

1.6.1.2 Terms from the Space Communications Cross Support Architecture Requirements Document

This Recommended Practice makes use of the following terms defined in reference [E6]:

a) Cross support service element;

b) Earth-space terminal;

e) Earth User node.

1.6.1.3 Definitions from Abstract Syntax Notation One

This Recommended Practice makes use of the following terms defined in reference [4]:

a) Abstract Syntax Notation One (ASN.1);

b) Object Identifier;

c) transfer syntax;

d) (data) type;

e) (data) value.

1.6.1.4 Terms from the Cross Support Transfer Services Specification Framework

This Recommended Practice makes use of the following terms defined in reference [1]:

a) acknowledgement (positive or negative);

b) confirmed operation;

c) Cross Support Complex;

d) Cross Support Service;

e) Cross Support Transfer Service;

f) Cross Support Transfer Service instance;

g) derivation;

h) Directive Identifier;

i) Event Identifier;

j) extension;

k) Functional Resource Instance;

l) Functional Resource Name;

m) Functional Resource Type;

n) invocation;

o) operation;

p) parameter;

q) Parameter Label;

r) performance;

s) procedure;

t) procedure configuration parameter;

u) production status;

v) protocol abort;

w) Published Identifier;

x) refinement;

y) response;

z) return;

aa) service instance provision period;

bb) service management parameter;

cc) service package;

dd) started procedure;

ee) stateful procedure;

ff) stateless procedure;

gg) three-phase operation;

hh) two-phase operation;

ii) unconfirmed operation.

1.6.1.5 Terms Defined in this Recommended Practice

1.6.1.5.1 abstract CSTS: A CSTS that is intended to serve as the basis for multiple concrete CSTSes that are to be created through refinement and/or extension of that abstract service.

1.6.1.5.2 prime procedure instance: The procedure type that reflects the primary purpose of the CSTS. A CSTS may comprise several instances of that procedure type, but only one of them can have the role of the prime procedure and is referred to as the prime procedure instance (see 3.1.2 and 3.1.6).

1.6.1.5.3 child CSTS: A CSTS that is derived from an existing CSTS referred to as the parent CSTS.

1.6.1.5.4 concrete CSTS: A CSTS that is intended to be directly implemented as specified.

NOTE – A concrete CSTS can also serve as the basis of further derived CSTSes by refinement and/or extension.

1.6.1.5.5 extended-only procedure: A procedure that has been derived from a parent procedure only by (a) adding new operations or (b) extending the operations used by the parent procedure.

1.6.1.5.6 parent CSTS: The CSTS from which others, the child CSTSes, are derived.

1.6.1.5.7 refined-and-extended procedure: A procedure that has been derived from a parent procedure by (a) adding new operations or extending the operations used by the parent procedure and by (b) modifying, e.g., narrowing, the parent procedure’s semantics or defining the derived procedure’s behavior or states in more detail than was done for the parent procedure.

1.6.1.5.8 refined-only procedure: A procedure that has been derived from a parent procedure only by (a) modifying, e.g., narrowing, the parent procedure’s semantics or by (b) defining the derived procedure’s behavior or states in more detail than was done for the parent procedure.

1.6.1.5.9 service-original procedure: A procedure that is defined specifically for a particular service on the basis of one or more of the operations specified in the CSTS Specification Framework (reference [1]). A service-original procedure is not derived from any other procedure.

1.6.1.5.10 started prime procedure instance: An instance of a started procedure that serves as the prime procedure instance of a CSTS.

1.6.2 NOMENCLATURE

1.6.2.1 Normative Text

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

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

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

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

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

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

1.6.2.2 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.

1.6.3 TYPOGRAPHIC AND OTHER CONVENTIONS

Typographic and other conventions used in this Recommended Practice are the same as those specified in 1.6.3.3 of reference [1].

1.7 REFERENCES

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

NOTE – A list of informative references is provided in annex E.

[1] *Cross Support Transfer Service—Specification Framework*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 921.1-B-1. Washington, D.C.: CCSDS, April 2017.

[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] *CCSDS Publications Manual*. Issue 4. CCSDS Record (Yellow Book), CCSDS A20.0-Y-4. Washington, D.C.: CCSDS, April 2014.

[4] *Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*. 5th ed. International Standard, ISO/IEC 8824-1:2015. Geneva: ISO, 2015.

2 OVERVIEW

CCSDS Cross Support Transfer Services (CSTSes) are based on two normative documents. The first document, the CSTS Specification Framework Recommended Standard (reference [1]), defines a set of standard *procedures*, each of which is comprised of one or more standard *operations*. These procedures are intended to form the basis of CSTSes that can serve a wide range of applications. However, the CSTS Specification Framework does not define how these procedures can be applied in order to create a particular CSTS. Furthermore, some future services may need procedures and/or operations with specialized capabilities beyond those found in the CSTS Specification Framework. While the CSTS Specification Framework does not address modification of procedures or operations specified therein for the purpose of creating new service types, it does, however, incorporate modification of procedures and operations to create new standard procedures as part of the CSTS Specification Framework itself. The methods by which these modifications can occur are the same as those used to modify procedures for the purposes of creating new services and are also illustrated in this Recommended Practice.

The second normative document for defining CSTSes, the Guidelines for the Specification of Cross Support Transfer Services (this Recommended Practice), defines (a) the requirements and constraints that govern the way in which a new conformant CSTS can be developed in accordance with reference [1]; and (b) guidance on the structure and content of real CSTS specifications.

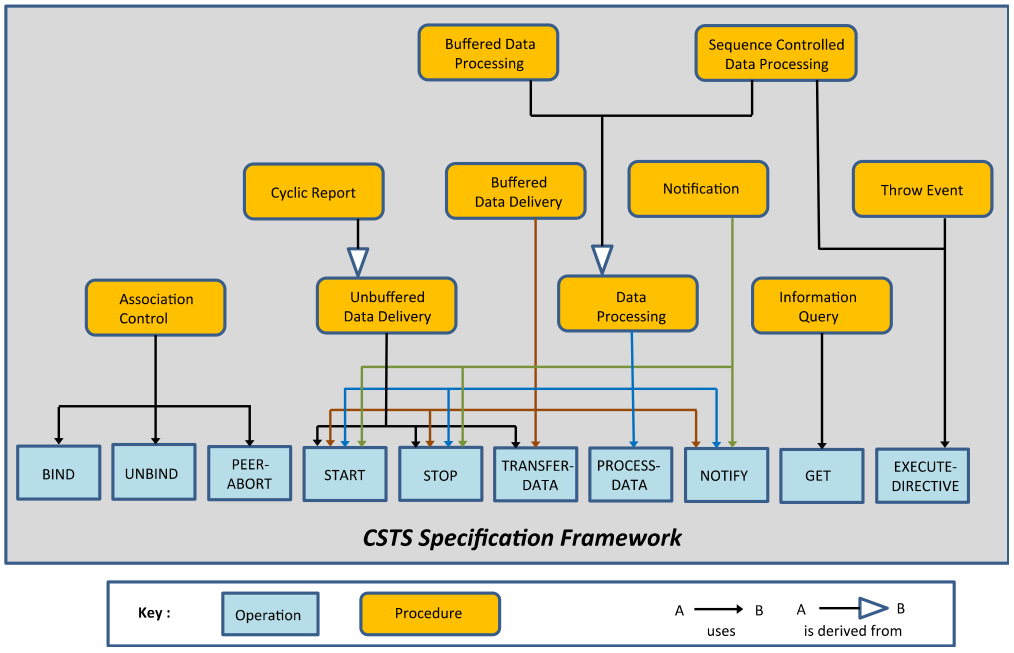

Figure 2-1: CSTS Framework Procedures and Operations

This Recommended Practice supports four major approaches for developing CSTSes from the CSTS Specification Framework procedures. Figure 2-1 graphically depicts the CSTS Specification Framework procedures and the operations that comprise them. As shown, one

CSTS Specification Framework procedure, Cyclic Report, is derived from the Unbuffered Data Delivery procedure, while the Buffered Data Processing and the Sequence-Controlled Data Processing procedures are derived from the (abstract) Data Processing procedure.

The first, and simplest, approach is to develop a service through direct adoption of one or more CSTS Specification Framework procedures without modification. This is the recommended approach if it can be made to work in satisfaction of the requirements of the application addressed by the new CSTS being specified. Figure 2-2 graphically depicts a new service (‘Service 1’) that is composed of unaltered CSTS Specification Framework procedures. In the figure, all of the standard CSTS Specification Framework procedures (round-cornered boxes) and the operations that comprise them (rectangles) are shown within the CSTS Specification Framework box. The hypothetical Service 1 is composed of the unmodified Association Control, Cyclic Report, and Sequence-Controlled Data Processing procedures.

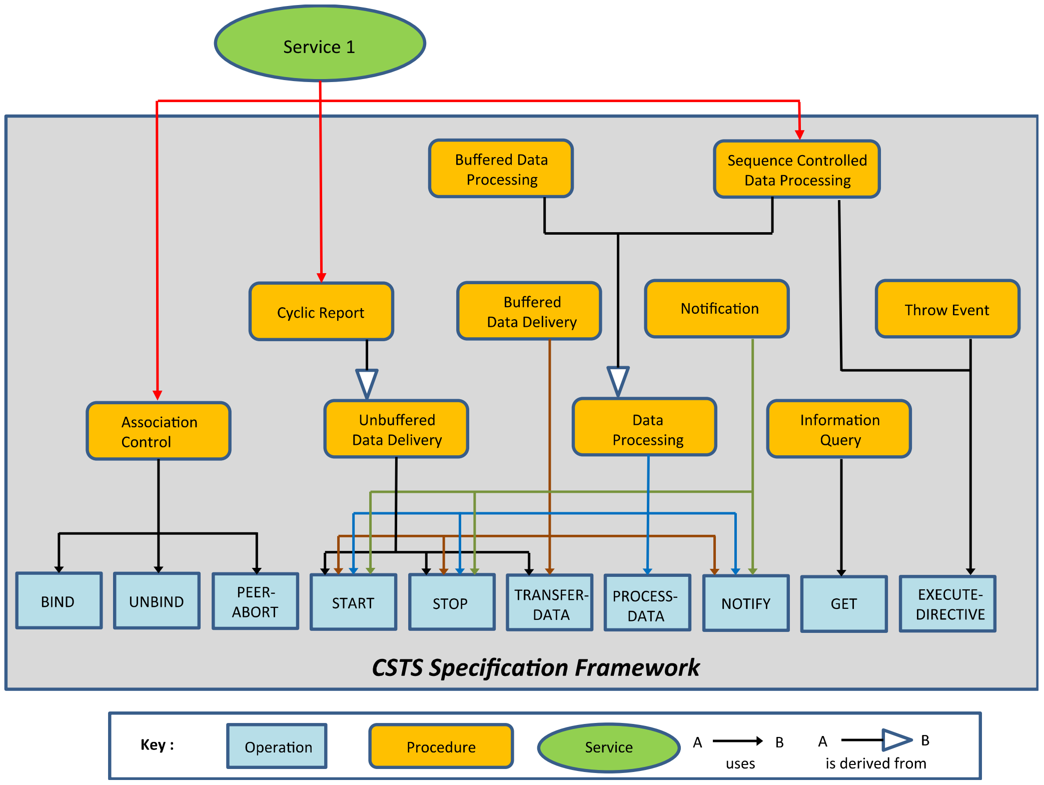

Figure 2-2: CSTS Directly Composed of CSTS Specification Framework Procedures

The specification of a CSTS that is composed solely of procedures as they are defined in the CSTS Specification Framework is the simplest kind of CSTS specification to write. The composition of the CSTS is essentially confined to the identification of the component standard procedures.

The second approach is to derive one or more procedures from existing CSTS Specification Framework procedures, in order to add or modify the capabilities of those procedures. Derivation of a procedure may occur through *refining* or *extending* the procedure (or both).

Refining an existing procedure can be accomplished by narrowing the range of the permitted value of a given parameters or by defining the behavior of the procedure to more detail.

Extending an existing procedure can be accomplished by adding parameters, adding behavior, and/or adding operations that exist in the CSTS Specification Framework but that do not belong to the existing procedure.

Figure 2-3 graphically depicts a new service (‘Service 2’) that is composed of the standard CSTS Specification Framework Association Control, Cyclic Report, and Sequence-Controlled Data Processing procedures plus a Configuration procedure that is derived from the Throw Event procedure by the addition of a NOTIFY operation.

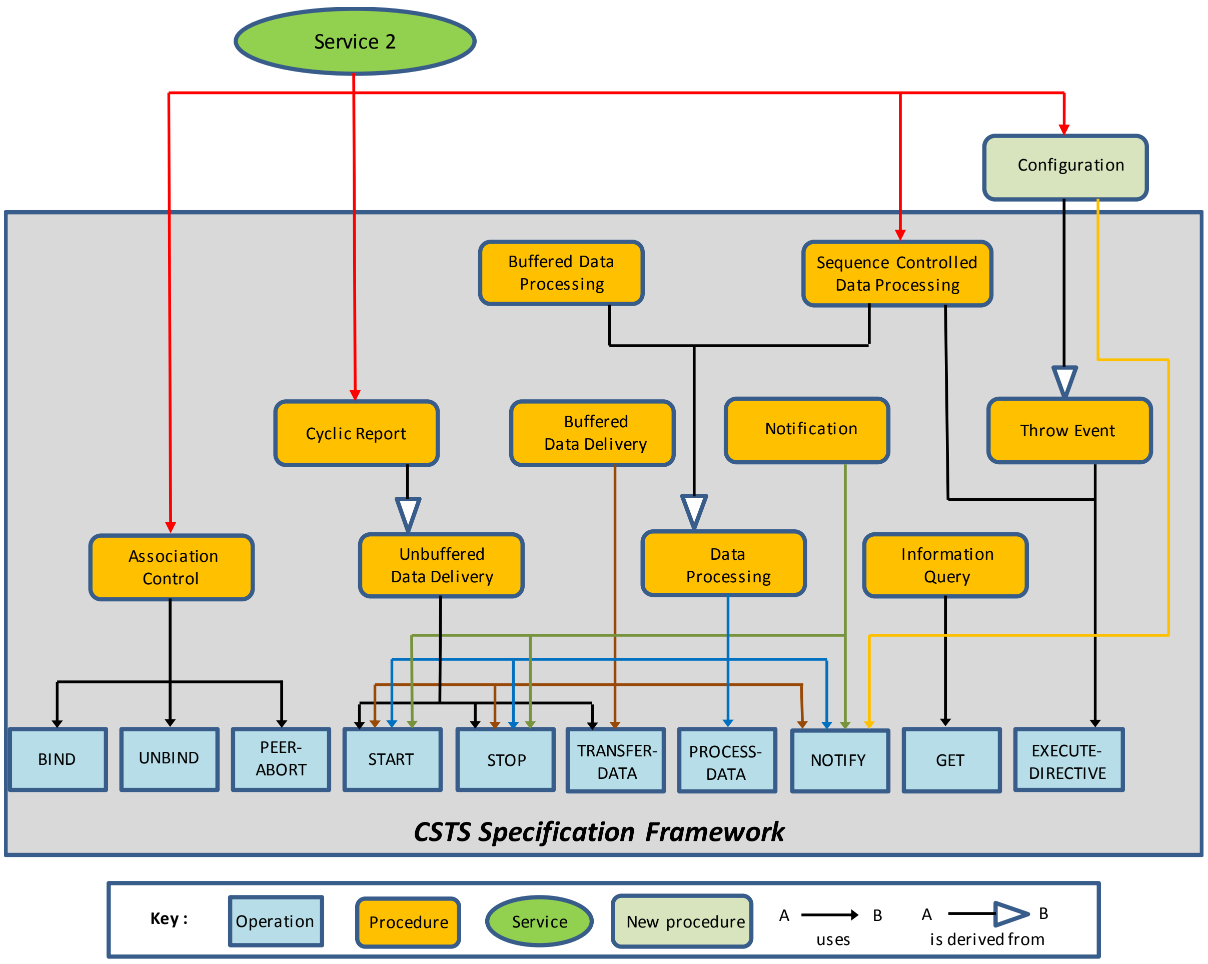

Figure 2-3: CSTS Composed of Derived and Standard CSTS Specification Framework Procedures

The specification of a CSTS that includes one or more derived procedures requires more information to be provided for each of the derived procedures. Specifically, for each derived procedure, the CSTS specification (a) defines how the behavior of the derived procedure differs from its parent procedure; (b) identifies which of the operations of the procedure are extended, refined, and/or new to the procedure (if any); and (c) defines the procedure state table.

The third approach to creating a new CSTS is to create one or more service-original procedures from the standard operations of the CSTS Specification Framework. These standard operations can be extended or refined to meet the needs of the service-original procedure. Figure 2-4 graphically depicts a new service (‘Service 3’) that is composed of the standard CSTS Specification Framework Association Control, Cyclic Report, and Sequence-Controlled Data Processing procedures, plus a service-original Notify & Get procedure that is composed of the START, STOP, NOTIFY and GET operations.

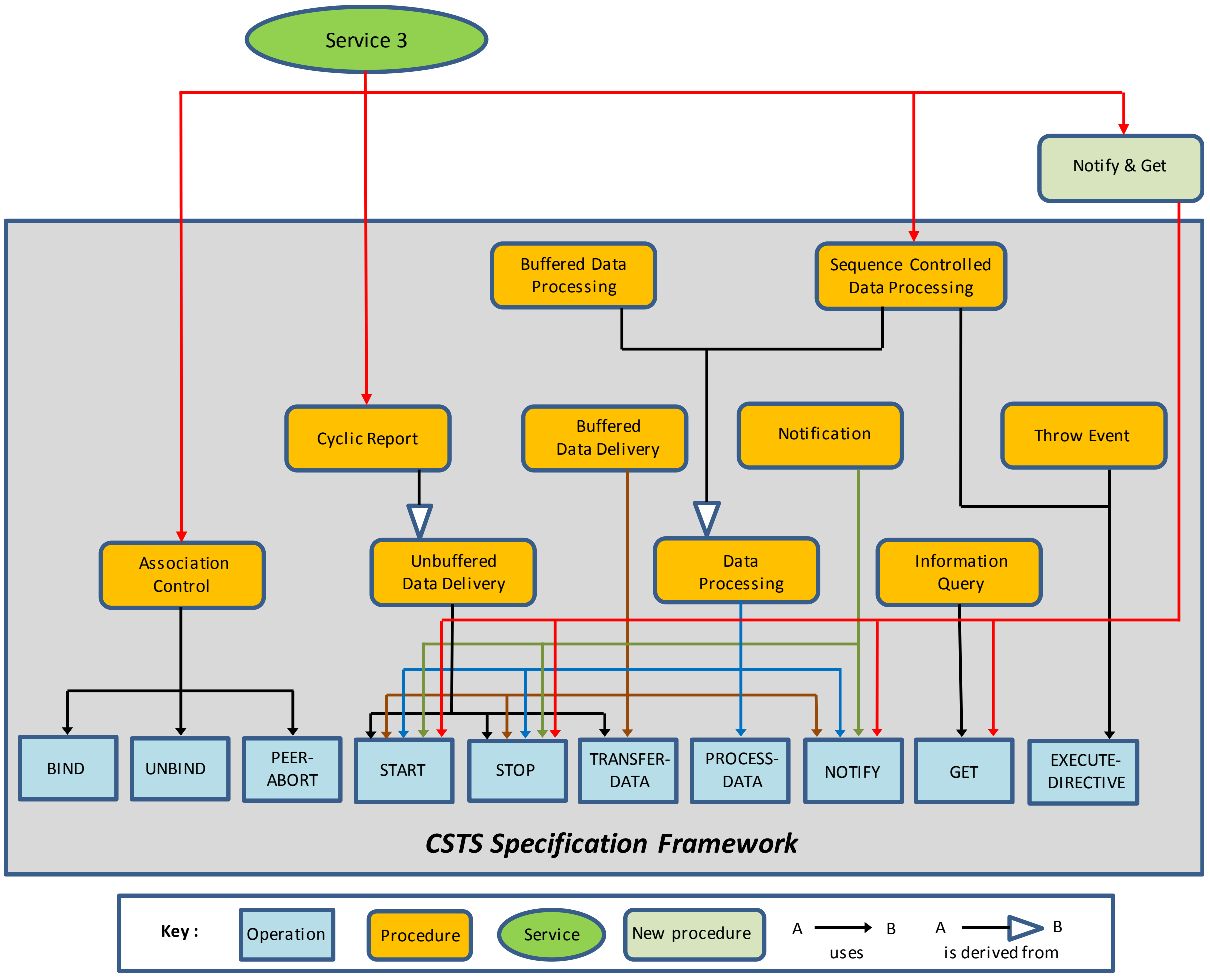

Figure 2-4: CSTS Composed of Service-Original and Standard CSTS Specification Framework Procedures

The specification of a CSTS that includes one or more service-original procedures requires that for each service-original procedure, the CSTS specification (a) defines the complete behavior of the procedure; (b) identifies which of the operations of the procedure are extended and/or refined; and (c) defines the procedure state table.

The fourth approach to creating a new CSTS is to derive it from an existing CSTS. A derived CSTS can be based on a *concrete* CSTS (that is, a CSTS that is fully capable of performing a useful service on its own) or an *abstract* CSTS (that is, a service that does not perform a useful service on its own, but rather is intended to serve as a common basis for a family of CSTSes that could be derived from it. Figure 2-5 graphically depicts a new Service 4 that is derived from an abstract Data Processing Service, which itself is composed of the CSTS Specification Framework Association Control, Cyclic Report, Sequence-Controlled Data Processing and the (derived) Configuration procedures.

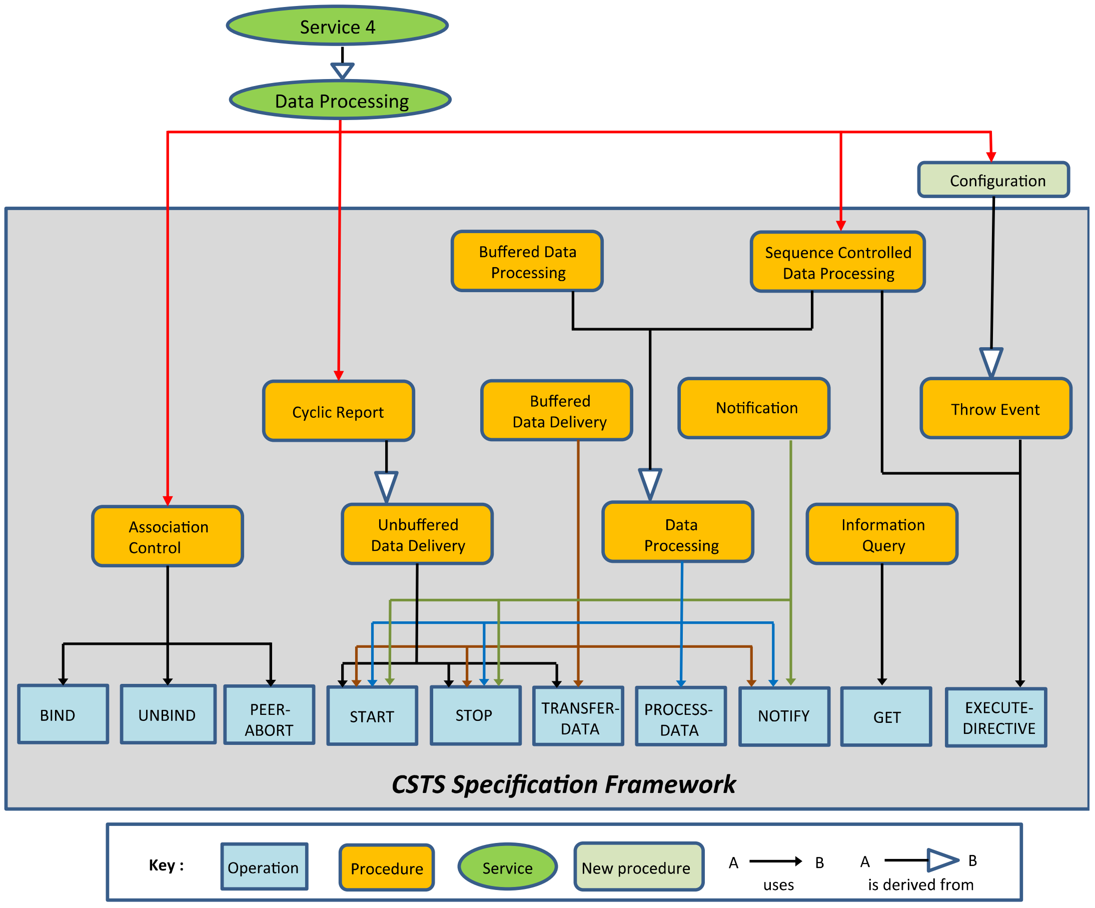

Figure 2-5: CSTS Derived from an Abstract CSTS

The specification of a CSTS that is derived from an existing CSTS can potentially involve all the other three methods of CSTS creation: direct inheritance of standard procedures (either through the parent service or as an added procedure); derivation of any of the procedures inherited from the parent CSTS or included from the CSTS Specification Framework; and addition of service-original procedures. Figure 2-6 illustrates the various derivation options.

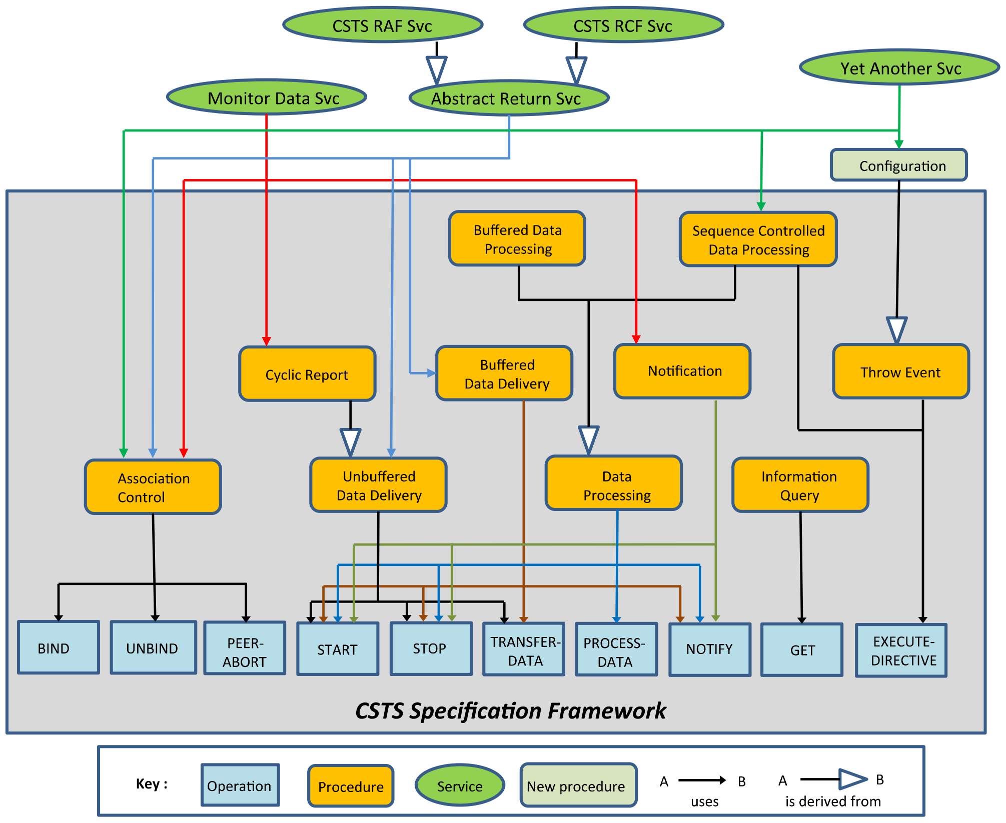

Figure 2-6: CSTS Derivation Examples

3 RULES FOR DEFINITION OF CROSS SUPPORT TRANSFER SERVICES

3.1 PROCEDURES OF THE CROSS SUPPORT TRANSFER SERVICES

3.1.1 A CSTS shall be defined in one of the following ways:

a) by direct adoption of the procedures defined in the CSTS Specification Framework, reference [1] (see figure 2-2);

b) by derivation from the procedures defined in reference [1] (see figure 2-3);

c) by creation of service-original procedures specific to the CSTS (see figure 2-4);

d) by derivation from a single existing CSTS. Requirements on the derivation from an existing CSTS are specified in 3.11.6 (see figure 2-5); and

e) by any combination of the above steps.

NOTES

1 If a service is derived from another service, the derived service is referred to as a *child* CSTS, and the service from which it is derived is referred to as the *parent* CSTS.

2 When specifying a new CSTS, an individual (service-original) procedure from an existing CSTS may be appropriate for direct adoption or further derivation by that new CSTS although the new CSTS cannot be derived from the existing CSTS. When a procedure that has been developed for a specific service is subsequently identified as having applicability to one or more other CSTSes, that procedure can be used by other CSTSes if such procedure is added to the CSTS Specification Framework (reference [1]) in the context of regular review of that document.

3.1.2 A CSTS shall comprise:

a) one Association Control procedure;

b) one and only one prime procedure instance; and

c) zero or more secondary procedure instances.

3.1.3 One of the procedures making up the CSTS shall have the capability to make the production status parameter value available.

NOTE – For example, this can be accomplished by including an instance of either the Cyclic Report or the Notification procedure.

3.1.4 An instance of any stateful or stateless procedure except the Association Control procedure may be designated as the prime procedure instance.

NOTE – As specified in subsection 2.5.3 of the CSTS Specification Framework (reference [1], the prime procedure (instance) of a service should reflect the primary purpose of the service. In practical terms it determines the state of the service.

3.1.5 All other instances of any stateful or stateless procedure except the Association Control procedure are secondary procedure instances.

NOTE – A CSTS can contain multiple secondary instances of the same procedure type. More details on multiple secondary instances of the same procedure type are provided in 3.8 and 3.9.

3.1.6 A CSTS may contain secondary instances of the same procedure type as that of the prime procedure instance.

3.1.7 A CSTS shall be specified as either a concrete or an abstract service.

3.2 CSTS STATE MACHINE

Each CSTS shall implement a state machine that complies with the requirements for service state tables specified in annex G of the CSTS Specification Framework (reference [1]).

3.3 CROSS SUPPORT TRANSFER SERVICE IDENTIFIERS

As part of the specification of a new CSTS type, a number of Object Identifiers (OID) need to be assigned. Annex subsection D5 of reference [1] specifies how the to be assigned OIDs are to be built and how the relevant OID subtree is structured. Annex subsection H2.3 of reference [1] defines the registration policy for these OIDs and the root node of the relevant subtree. The SANA registry containing the overall CCSDS OID tree may be found at http://sanaregistry.org/r/oid/oid.html.

3.4 ASSOCIATION CONTROL PROCEDURE

3.4.1 A CSTS should directly adopt the Association Control procedure defined in either the CSTS Specification Framework (reference [1]) or (for a child CSTS) the parent CSTS with no further extension or refinement, if at all possible.

3.4.2 If the requirements of a CSTS are such that the Association Control procedure as defined in the CSTS Specification Framework (reference [1]) or the parent CSTS cannot be directly adopted, a CSTS may derive an Association Control procedure that is an extension or refinement of the Association Control procedure defined in 4.3 of the CSTS Specification Framework ([1]).

NOTE – Caution is advised in deriving a procedure from the Association Control procedure. Subsection 4.3 of the CSTS Specification Framework (reference [1]) specifies constraints on the extensibility of the Association Control procedure.

3.5 DIRECT ADOPTION OF PROCEDURES

3.5.1 A CSTS may directly adopt a procedure from the CSTS Specification Framework (reference [1]) or, if it is a derived service, from its parent CSTS.

3.5.2 A CSTS should directly adopt procedures to the greatest extent possible.

3.5.3 A CSTS may directly adopt a procedure only if no new parameters need to be added to any of the operations of the procedure.

3.5.4 A CSTS may directly adopt a procedure only if none of the extensible parameters of any of the operations of the procedure need to be extended.

3.5.5 A concrete CSTS may directly adopt a procedure with abstract parameters (refer to B6) only if every abstract parameter is resolved.

NOTE – An example of how an abstract parameter could be resolved is as follows: If the CSTS uses the CSTS Specification Framework Buffered Data Delivery or Unbuffered Data Delivery procedure, the `data` parameter of the TRANSFER-DATA invocation can be resolved either by (a) refining the value of the opaqueString choice of the `data` parameter or by (b) defining an extension type for the extendedData choice of the `data` parameter of the common TransferDataInvocation (refer to the `AbstractChoice` type specified in F4.3 of reference [1]).

3.5.6 An abstract CSTS may directly adopt a procedure with abstract parameters.

3.5.7 A concrete CSTS may directly adopt a procedure only if the behavior of the procedure is completely specified.

NOTE – An example of behavior that is not completely specified is the following statement:

The processing performed by the provider in response to the STOP invocation is to be specified by the service using the procedure.

3.5.8 An abstract CSTS may directly adopt a procedure even if the behavior is not completely specified.

3.6 DERIVATION OF PROCEDURES

3.6.1 If there is no standard procedure that can be directly adopted either from the CSTS Specification Framework (reference [1]) or (for a child CSTS) from the parent CSTS, a procedure that is derived from an existing procedure should be used in the specification of the CSTS whenever possible. Derivation of a procedure consists of refinement and/or extension of an existing procedure.

NOTE – When an existing standard procedure meets the requirements, deriving a new procedure for a service is discouraged.

3.6.2 A CSTS may refine a procedure through:

a) narrowing the specification of the values of one or more parameters of constituent operations of the parent procedure;

b) narrowing the specification of the behavior of the parent procedure; and/or

c) defining the states of the parent procedure in more detail.

3.6.3 A CSTS may extend a procedure through:

a) adding operations; and/or

b) extending the used operations.

3.7 DEFINITION OF SERVICE-ORIGINAL PROCEDURES

3.7.1 If there is no standard procedure available in the CSTS Specification Framework (reference [1]) or the parent CSTS (in the case of a child CSTS) that can either be directly adopted or from which a procedure can be derived to meet the requirements of the service, a procedure that is specific to the service may be created from any of the standard operations of the CSTS Specification Framework except BIND, UNBIND, and PEER-ABORT (which are reserved for the Association Control procedure).

NOTE – When an existing standard procedure or derived procedure meets the requirements of a service, defining a new service-original procedure for that service is discouraged.

3.7.2 A CSTS may include one or more new procedures that are created from previously defined standard operations.

3.7.3 If a CSTS service-original procedure has a START operation, it must have a STOP operation.

3.7.4 If a CSTS service-original procedure has a START and a STOP operation, it shall be a stateful procedure.

3.7.5 If a CSTS service-original procedure has a three-phase operation, it shall be a stateful procedure.

NOTES

1 It is possible for a procedure to be stateful but have neither a START/STOP pair of operations nor a three-phase operation.

2 It is possible for a procedure to contain both a START/STOP pair of operations and a three-phase operation. For such a procedure the START and STOP operations control the state of the procedure.

3.7.6 The following must be defined for each service-original procedure:

a) the behavior of the procedure, including (but not limited to)

   1) how it becomes active and inactive, in case the procedure is stateful;

   2) the legal sequence of and relationships among the operations that may be invoked as part of the procedure; and

   3) how the procedure terminates (including but not limited to the effects of protocol abort, PEER-ABORT, or UNBIND).

b) the operations from the CSTS Specification Framework (reference [1]) that comprise the procedure, where each operation may be directly adopted, refined, or extended;

c) the (provider) state table of the procedure, in case the procedure is stateful.

3.8 INSTANCES OF A PROCEDURE TYPE

3.8.1 Any procedure type (other than the Association Control procedure) may be used for:

a) the prime procedure instance;

b) one or more instances of a secondary procedure; or

c) the prime procedure instance and one or more instances of a secondary procedure.

3.8.2 The minimum number of secondary instances of each procedure type that must be supported by every implementation of the service shall be specified.

3.8.3 The maximum number of secondary instances of each procedure type may be specified for the service, or the maximum number may be left unbounded.

NOTES

1 The maximum number of instances of a given procedure type is the upper bound of the number of instances that may be enabled by any implementation of the service.

2 The primary and all secondary procedure instances of a Cross Support Transfer Service instance are assumed to be enabled as part of the binding of that service instance.

3.9 OPTIONAL PROCEDURES

3.9.1 Any procedure that is not used as the prime procedure may be defined as an optional secondary procedure for a CSTS, where optional means that an implementation of the service is not required to support the procedure.

3.9.2 An optional secondary procedure for a CSTS shall have a minimum number of secondary procedure instances set to zero.

3.9.3 An optional secondary procedure for a CSTS may have any number of maximum number of instances of that procedure (including unbounded).

3.9.4 If an implementation of a CSTS does not support an optional secondary procedure of that service, the implementation shall respond to an invocation of a two-phase or three-phase operation of that procedure by returning a negative response with the `diagnostic` parameter set to ‘unsupported option’.

NOTE – This implies that if an optional procedure is not implemented, the implementation still has enough knowledge of that procedure to recognize the invocation of its operations and reject such invocation properly, instead of treating it as the invocation of an unknown operation that would result in the invocation of PEER-ABORT.

3.10 PROCEDURE IDENTIFIER FOR AN EXTENDED OR SERVICE-ORIGINAL PROCEDURE

As part of the specification of a new procedure, be it derived from an existing procedure or be it a service-original procedure, a number of Object Identifiers (OIDs) need to be assigned. Annex subsection D4 of reference [1] specifies how the to be assigned OIDs are to be built and how the relevant OID subtrees are structured. Annex subsection H2.2 of reference [1] defines the registration policy for these OIDs.

3.11 DERIVATION OF OPERATIONS

3.11.1 OVERVIEW

Derivation is a mechanism that allows extending or refining an operation by means of one or more of the following options:

a) refining values of parameters specified in the parent procedure;

b) extending parameters specified in the parent procedure;

c) specifying the syntax and semantic of abstract parameters;

d) extending parameters for an abstract CSTS;

e) extension of an operation belonging to a procedure of the parent CSTS.

NOTE – Adding new parameters or additional content to existing parameters is called *extension*. Constraining the values of parameters is called *refinement*. This section specifies the requirements for and constraints on the extension and refinement of operation parameters.

3.11.2 REFINEMENT OF PARAMETER VALUES

3.11.2.1 The refinement of a parameter must conform to the specification of the parent parameter.

NOTE – That is, the refined parameter values always need to be a subset of the parameter values specified for the parent operation.

Example 1: If in the parent CSTS an octet-string valued parameter is refined to be the name of a city, the child CSTS can refine that parameter to be the name of a city in North America, but it cannot refine it to be the name of an automobile brand.

Example 2: If in the parent operation the extension parameter `day` is given the type `DaysOfWeek`, and a (first generation) child operation refines `day` to be only the weekend days, the `day` parameter of a (second generation) child operation of that (first generation) child operation cannot be 'refined' to include any of the days Monday through Friday.

3.11.3 EXTENDING OPERATION PARAMETERS

3.11.3.1 Overview

A CSTS can extend an operation through:

a) adding new parameters to the invocation or response message;

b) extending the range of values for already existing parameters;

c) changing an unconfirmed operation into a confirmed operation.

Operations can be extended at explicitly defined extension points. The extension points for the common CSTS operations are defined in the CSTS Specification Framework (reference [1]). Individual CSTSes can define their own extension points, but only by extending the extension points that are made available to them in the operations that they adopt or derive from their sources (i.e., a common CSTS Specification Framework operation or, in the case of a child CSTS, from the operation of the parent CSTS) in a way that allows further extension.

For a CSTS that is created from the CSTS Specification Framework (reference [1]), any operation of a CSTS can be extended through the addition of one or more parameters to the invocation, positive result return, negative result return, positive result acknowledgement, or negative result acknowledgement (as appropriate to and available for that operation; see 3.11.3.2). For a CSTS that is derived from another CSTS, any operation of the CSTS can be extended unless explicitly prohibited by the parent CSTS.

For a CSTS that is created from the CSTS Specification Framework (reference [1]), the values of the `diagnostic` component of the `result` parameter of the negative return or negative acknowledgement can be extended for any operation (see 3.11.3.3). For a CSTS that is derived from another CSTS, values of the `diagnostic` component of the `result` parameter of any operation of the CSTS can be extended unless explicitly prohibited by the parent CSTS.

For a CSTS that is created from the CSTS Specification Framework (reference [1]), the values of the `event-name` parameter of the invocation can be extended for any operation derived from the NOTIFY operation (see 3.11.3.4). For a CSTS that is derived from another CSTS, values of the `event-name` parameter of any NOTIFY operation of the CSTS can be extended unless explicitly prohibited by the parent CSTS.

For a CSTS that is created from the CSTS Specification Framework (reference [1]) procedures, the types of valid values for the `qualified-parameters` parameter of the GET positive return can be extended for any operation derived from the GET operation (see 3.11.3.5). The `data` parameter of the TRANSFER-DATA invocation of the Cyclic Report procedure can be extended for any operation derived from the TRANSFER-DATA operation (see 3.11.3.5) within the constraints of the refinement to the `CyclicReportTransferDataInvocDataRef` type as specified for the TRANSFER-DATA operation used by the Cyclic Report procedure. For a CSTS that is derived from another CSTS, value types of the `qualified-parameters` parameter of any operation of the CSTS can be extended unless explicitly prohibited by the parent CSTS.

The specification of a CSTS can explicitly prohibit the addition of parameters, `diagnostic` values, `event-name` values, and/or `qualified-parameters` value types in any service that is derived from that CSTS.

Operation extension parameters, `diagnostic` values extensions, `event-name` value extensions, and `qualified-parameters` value type extensions each have a syntax and a transfer syntax. Each of these syntax/transfer syntax pairs has a syntax identifier that allows the CSTS user and provider to interpret the contents of the extensions.

The specification of a CSTS can extend

a) an unconfirmed operation into a confirmed operation. The extension is performed by adding a return to the unconfirmed operation;

b) a confirmed operation into three-phase operation. The extension is performed by adding an acknowledgement to the confirmed operation.

NOTE – Operations extended as outlined above can specify the use of the newly defined return or acknowledgement PDUs to be optional. In that case any procedure using such operation has to specify if the optional PDU is required to be used or not. The PROCESS-DATA operation and its use by the Data Processing procedure and by the Sequence-Controlled Data Processing procedure as specified in reference [1] are examples of this approach.

3.11.3.2 Adding Extension Parameters to Operation Invocations, Acknowledgements, and Returns

3.11.3.2.1 Extension parameters may be added to the invocation of an operation.

3.11.3.2.2 Extension parameters shall be added to the invocation of an operation at the extension points for additional parameters defined in annex F of the CSTS Specification Framework (reference [1]) and as further specified in B2.1.

NOTE – Annex subsection B2.1 identifies the extension points for adding new parameters to the invocations of common operations and of procedure-specific operations that have added parameters.

3.11.3.2.3 Extension parameters may be added to the response, i.e., the return or acknowledgement of an operation that are applicable only when the `result` parameter has the value ‘positive’.

3.11.3.2.4 Extension parameters applicable in case of positive responses shall be added to the return or acknowledgement of an operation at the extension points of positive responses defined in annex F of the CSTS Specification Framework (reference [1]) and as further specified in B2.2.1.

NOTE – Annex subsection B2.2.1 identifies the extension points for adding new parameters to positive returns or acknowledgements of common operations and of procedure-specific operations that have added parameters.

3.11.3.2.5 Extension parameters may be added to the response, i.e., the return or acknowledgement of an operation that are applicable only when the `result` parameter has the value ‘negative’.

3.11.3.2.6 Extension parameters applicable in case of negative responses shall be added to the return or acknowledgement of an operation at the extension points of negative responses defined in annex F of the CSTS Specification Framework (reference [1]) and as further specified in B2.2.2.

NOTE – Annex subsection B2.2.2 identifies the extension points for adding new parameters to the negative returns and acknowledgements of common operations and of procedure-specific operations that have added parameters.

3.11.3.2.7 Specification of extension parameters for the invocation, positive return, negative return, positive acknowledgement, or negative acknowledgement of an operation of a CSTS shall include the following:

a) each extension parameter shall be identified (named), along with its respective type, value range, and engineering units;

b) the syntax and the transfer syntax (see reference [1]) of the new parameters shall be defined;

c) if the CSTS does not prohibit extension parameters for services that may be derived from the CSTS, the syntax and transfer syntax shall be constructed in a way that allows further extension;

d) each new parameter shall be defined with respect to applicability to the operation invocation, acknowledgement, or return; and

e) the syntax and transfer syntax of the parameters extension shall be assigned a unique syntax identifier under the extended service parameters subtree (see 4.10.5.6 and 4.10.5.7) for that CSTS.

NOTE – The derived service specification example in annex C illustrates the specification of the syntax and transfer syntax for extension parameters for an operation invocation.

3.11.3.3 `diagnostic` Extension

3.11.3.3.1 If the CSTS derives an operation from the CSTS Specification Framework that has a return or acknowledgement, additional `diagnostic` values may be added to the negative response.

3.11.3.3.2 Additional `diagnostic` values shall be added to the negative response extension points for `diagnostic` values defined in annex F of the CSTS Specification Framework (reference [1]) and further specified in B3 of this Recommended Practice.

NOTE – Annex subsection B3 identifies the extension points for the `diagnostic` parameter values of the common operations and the procedure-specific operations that extend the `diagnostic` values.

3.11.3.3.3 Specification of `diagnostic` extension values shall include the following:

a) each new value shall be identified (named);

b) the syntax and transfer syntax of the new values shall be defined;

c) if the CSTS does not prohibit extension of the `diagnostic` values in services that may be derived from the CSTS, the syntax and transfer syntax shall be constructed in a way that allows further extension;

d) each new `diagnostic` value must be specified as to which negative result it pertains: acknowledgement or return;

e) the syntax and transfer syntax of the `diagnostic` values extension shall be assigned a unique syntax identifier under the extended service parameters subtree (see 4.10.5.6 and 4.10.5.7) for that CSTS.

3.11.3.4 `event-name` Extension

3.11.3.4.1 If the CSTS creates a procedure that uses the NOTIFY operation or adds a NOTIFY operation to a procedure, additional values of the eventId component of the `event-name` parameter may be added to the set of events associated with the given procedure.

3.11.3.4.2 The Published Identifiers of the events added to the <DEF> procedure shall be registered under the ‘p<DEF>eventsId’ subbranch as specified in annex D of the CSTS Specification Framework (reference [1]) and further specified in B4.

NOTES

1 Annex subsection B4 identifies the extension points for the `event-name` parameters of the common NOTIFY operation and the CSTS Specification Framework procedures that include the NOTIFY operation.

2 According to the CSTS Specification Framework (reference [1]) the `event-name` is composed of an `fRorProcedurename` (in form of a Functional Resource Name or a Procedure Instance Identifier and an Event Identifier—refer to F4.3 of reference [1]). Depending on the event, an `event-value` can be specified in addition. The `event-name` argument is also referred to as ‘published event’.

3.11.3.4.3 Specification of `event-name` extension values shall include the following:

a) each new event shall be identified (named) by means of an associated Object Identifier registered as specified in 3.11.3.4.2;

b) the syntax and transfer syntax of the associated `event-value` shall be defined, which may be set to ‘empty’ in case no `event-value` needs to be reported for the given event;

c) if the CSTS does not prohibit adding of new events and associated `event-value` parameters in services that may be derived from the CSTS, the syntax and transfer syntax of `event-value` shall be constructed in a way that allows further extension.

3.11.3.4.4 If the CSTS creates a procedure that uses the NOTIFY operation or adds a NOTIFY operation to a procedure, or uses a CSTS Specification Framework procedure that contains a NOTIFY operation, additional content may be added to any or all of the published events by extending the NOTIFY invocation with additional parameters.

NOTE – Subsection 4.6.4.2 of the CSTS Specification Framework (reference [1]) is an example of such extension.

3.11.3.4.5 If required, additional content shall be added to any of the `event-name` choices (see 3.11.3.4.4) of a NOTIFY operation invocation for a CSTS by extending the `event-value` parameter.

3.11.3.5 `qualified-parameters` Extension

3.11.3.5.1 If the CSTS uses an operation from the CSTS Specification Framework that has a parameter of the `QualifiedParameter` type, additional parameter types may be added to that parameter (see annex C of the CSTS Specification Framework (reference [1]).

3.11.3.5.2 Additional parameter types shall be added at the extension point for the parameter types of type `QualifiedParameter` defined in annex F of the CSTS Specification Framework (reference [1]) and further specified in B5 of this Recommended Practice.

NOTE – Annex subsection B5 identifies the extension points for the `qualified-parameters` parameters of the common GET operation and the CSTS Specification Framework Cyclic Report procedure.

3.11.3.5.3 Specification of parameter type extension values shall include the following:

a) each new parameter type shall be identified (named);

b) the syntax and transfer syntax of the new parameter type(s) shall be defined;

c) if the CSTS does not prohibit extension of the valid value types in services that may be derived from the CSTS, the syntax and transfer syntax shall be constructed in a way that allows further extension; and

d) the syntax and transfer syntax of the valid value types extension shall be assigned a unique syntax identifier under the <xyz>ExtendedServiceParameters subtree (see annex subsection D5 of the CSTS Specification Framework, reference [1]) for that CSTS.

3.11.4 ABSTRACT PARAMETERS

3.11.4.1 Overview

A parameter of an operation is abstract if it has no specific type of its own and requires that the procedure that uses that operation specifies the type before it can be used in a concrete service.

NOTES

1 The CSTS Specification Framework (reference [1]) uses the type `AbstractChoice` for abstract parameters as, e.g., for the abstract `data` parameter of the TRANSFER-DATA operation.

```
AbstractChoice ::= CHOICE
{ opaqueString [0] OCTET STRING
, extendedData [1] Embedded
}
```

2 The CSTS Specification Framework uses the abstract `data` parameter also in the PROCESS-DATA operation.

3 The CSTS Specification Framework does not use the term 'abstract data parameter'. The term is introduced in these Guidelines as a name for the generalization of the `data` parameters of the PROCESS-DATA and TRANSFER-DATA operations.

3.11.4.2 An abstract `data` parameter shall be resolved in one of two ways:

a) by specifying the parameter to be an octet-string that is refined to be of a specific format (type); or

b) by extending the parameter to contain content of a specified type.

3.11.4.3 An abstract `data` parameter should be resolved by refinement as a typed octet string when the content is either (a) simple or (b) is complex but has no effect on the behavior of the procedure.

NOTE – For example, a service that nominally transfers CCSDS Version 1 Transfer Frames but in actuality does not act on any field value within those transfer frames (i.e., the behavior is unaffected by the refined content) can be considered. An abstract parameter that is derived for this service to contain a CCSDS Version 1 Transfer Frame would be resolved as 'an octet string formatted as a CCSDS Version 1 Transfer Frame'.

3.11.4.4 An abstract `data` parameter should be resolved by extension when the content is (a) of a complex type; and (b) the behavior of the procedure depends on the content of extension.

NOTES

1 For example, an abstract parameter that is derived to contain a set of variable-length CCSDS space packets, each of which is processed individually by the procedure, can be considered. Because the behavior of the procedure depends on the content of the extension (in order to separate the individual packets), the parameter would be extended as a complex data structure that explicitly demarcates the individual packets.

2 Extension of a parameter requires derivation of a new procedure type and specification of a syntax for that parameter extension. Refinement of a parameter does not require derivation of a new procedure if the refinement has no effect on the behavior of the provision of the service. Therefore it is preferable to refine an abstract procedure rather than extend it if derivation of a new procedure can be avoided.

3.11.4.5 If a procedure that contains any abstract `data` parameters is to be used for a concrete CSTS, each of those abstract `data` parameters must be resolved by that CSTS specification. That is, each of those abstract `data` parameters must be specified as either a refined octet string or having an extended value type.

NOTE – Abstract `data` parameters can remain abstract in abstract CSTSes.

3.11.4.6 If an abstract `data` parameter is resolved by extension:

a) the extended value type shall be identified (named);

b) the type, value range, and engineering units of the extended value shall be specified;

c) the syntax and the transfer syntax of the extension content shall be defined;

d) the syntax and transfer syntax shall be constructed in a way that allows further extension, provided the CSTS does not prohibit extended content for services that may be derived from that CSTS; and

e) the syntax and transfer syntax of the `data` parameter extension shall be assigned a unique syntax identifier under the extended service parameters subtree (see annex subsection D5 of the CSTS Specification Framework (reference [1])) for that CSTS.

3.11.4.7 Once an abstract `data` parameter of an operation is resolved as either a refined octet string or an extended data type, any further derivation of that parameter shall remain in the context of that refinement or extension.

NOTE – For example, the `data` parameter of the Unbuffered Data Delivery procedure TRANSFER-DATA operation is abstract. The Cyclic Report procedure is derived from the Data Delivery procedure. The `data` parameter of the Cyclic Report TRANSFER-DATA operation is derived from the `data` parameter of the Unbuffered Data Delivery TRANSFER-DATA operation by defining the parameter through extension as the `qualified-parameters` parameter (see reference [1]). Any derivation of the Cyclic Report TRANSFER-DATA `data` parameter is therefore constrained to extend the `qualified-parameters` parameter or add a new extension parameter. A CSTS using the Cyclic Report procedure cannot revert to using the opaque octet-string option that is available in the `data` parameter of the Unbuffered Data Delivery procedure.

3.11.4.8 An abstract Directive Identifier of the EXECUTE-DIRECTIVE operation is resolved by specifying the Object Identifiers associated with the directives required by the procedure.

NOTE – The Sequence-Controlled Data Processing procedure specified in 4.8 of the CSTS Specification Framework (reference 1) provides an example by introducing the new 'reset' directive.

3.11.5 PARAMETER EXTENSION FOR AN ABSTRACT CSTS

If the CSTS is abstract, operations for that abstract CSTS may add one or more new parameters, new `diagnostic` values, new `event-name` values, or `qualified-parameters` value types while deferring the specification of the types, value ranges, and/or syntax for any of those parameters or values to the services that are derived from it.

NOTE – Any concrete service that is derived from an abstract service with incompletely defined extension parameters will be required to complete the definition of those parameters.

3.11.6 OPERATION EXTENSION

3.11.6.1 Discussion

The definition of the new service allows the extension of the used operation such that

a) an unconfirmed operation is converted into a confirmed, i.e., two-phase operation that has a return;

b) a confirmed two-phase operation is converted into three-phase operation, i.e., an operation that has an acknowledgement and a return.

3.11.6.2 Identification of Parameters

The specification shall clearly identify the mandatory/conditional parameters used for the return and, if applicable, for the acknowledgement.

3.12 DERIVING A NEW SERVICE FROM AN EXISTING CROSS SUPPORT TRANSFER SERVICE

Deriving a child CSTS from a parent CSTS shall conform to the requirements for creating a new CSTS as defined in 3.3 through 3.11.3, with the following additional requirements:

a) A child CSTS shall contain at least all of the procedure types of the parent CSTS.

NOTES

1 In addition to the procedure types of the parent CSTS, a child CSTS can add an existing CSTS Specification Framework procedure. The addition is made by adopting the procedure as is or deriving a new procedure.

2 The child CSTS can create a new procedure making use of the parent CSTS operations.

3 A child CSTS may refine or extend operation parameters of the procedures (including the procedure type of the prime procedure instance).

b) A child CSTS shall implement the same procedure type for the prime procedure instance as its parent CSTS, where, however, the prime procedure may be extended.

3.13 BACKGROUND—CONFIGURATION OF CROSS SUPPORT TRANSFER SERVICES

Any CSTS will be characterized by parameters, some of which will be fixed and specified as part of the definition of the service, and some of which referred to as configuration parameters will have to be configured. The configuration parameters of a CSTS may be set through Service Management, service provision, or some combination thereof. If at least the initial value of a configuration is set by Service Management, the parameter is referred to as service management parameter.

Service Management sets configuration parameters of a CSTS in an out-of-band manner with respect to the provision of that service. The reasons for using Service Management to set configuration parameters of a CSTS include:

a) establishing values (or subranges or subsets of values) that are essentially permanent for all instances of a given service type within the context of a given CSTS relationship. Typically these values/ subranges/ subsets correspond to the intersection of the capabilities of the provider and the capabilities/needs of the user which do not normally change;

b) establishing values at the time that a Cross Support Transfer Service instance is scheduled, which may be days or weeks before the provision period for that service instance. Typically, configuring parameter values is performed as part of scheduling to validate that the configuration is valid and supportable with sufficient lead time to make alternate arrangements if the requested configuration cannot be provided, e.g., because of violation of the bounds imposed by the Service Agreement or anticipated lack of required resources during the proposed service instance provision period.

For some services, the values of some configurable parameters may not be knowable when the Cross Support Transfer Service instance is scheduled (or earlier), or some configurable parameter values may need to be resettable while the service instance is executing. In such cases, values of configurable parameters can be set as part of the provision of the service itself.

4 STRUCTURE AND CONTENT OF A CSTS SPECIFICATION

4.1 OVERVIEW

This section specifies the rules for the structure and content of a specification of a CSTS.

4.2 FRONT COVER AND FRONT MATTER

4.2.1 The front cover and front matter of the CSTS Specification shall conform to the format and content specified in reference [3].

4.2.2 The title of the CSTS specification shall be of the form 'Cross Support Transfer Services **—** <name of data that is transferred> Service'.

4.2.3 The CSTS specification shall be registered with the CCSDS Secretariat as a member of the Cross Support Transfer Service series of specifications.

4.3 'INTRODUCTION' SECTION

4.3.1 The CSTS specification shall contain an informative section 1 entitled 'Introduction', which shall conform to the format and content for that section specified in [3], with additions and qualifications as specified in the following clauses.

4.3.2 The 'Scope' subsection shall state that the service is defined in the context of the CSTS Specification Framework (reference [1]) and in accordance with the Guidelines (this Recommended Practice), with accompanying references (see 4.3.9).

4.3.3 For a service that is derived from an existing CSTS, the 'Scope' subsection shall include a statement that the service is derived from the (named) parent service, with accompanying reference.

4.3.4 The 'Applicability' subsection should include the particular characteristics of the service that make use of the CSTS Specification Framework (reference [1]) appropriate.

NOTE – For example, a need for reliable transfer of the application data is reason for applicability of any service developed using the CSTS Specification Framework. Other statements of applicability can derive from the specific CSTS procedures that are used by the service, e.g., a service that employs the Buffered Data Delivery procedure would include an applicability statement such as 'the XYZ service is applicable when the data must be retrievable not only as it is generated but for some (service-specific) time period after it has been generated'.

4.3.5 The contents of the 'Rationale' and 'Document Structure' subsections shall be in accordance with reference [3].

4.3.6 Under the 'Definitions' subsection, the 'Terms' subsection shall include:

a) if applicable, a subsection entitled ‘Terms defined in the CSTS Specification Framework, reference [<reference number>]’, containing the list of all terms that are defined in the CSTS Specification Framework that are explicitly used in the CSTS specification; this list shall contain all terms that are listed in the ‘Definitions’ subsection 1.6.1 of the CSTS Specification Framework (reference [1]), even if those terms are originally defined elsewhere;

b) as applicable, for every other document that defines terms that are explicitly used in the CSTS, a subsection entitled ‘Terms defined in <document name>, reference [<reference number>]’; for a service derived from an existing CSTS, this includes terms from the parent CSTS that are explicitly used in the specification;

c) if applicable, a subsection entitled ‘Terms defined in this specification’, containing the list of terms that are specific to the document, along with their definitions.

4.3.7 If the CSTS specification uses only the conventions specified in the CSTS Specification Framework (reference [1]), the ‘Conventions’ subsection shall consist of the statement:

‘This Recommended Standard uses the conventions defined in the CSTS Specification Framework (reference [<reference number>])’.

4.3.8 If the CSTS specification uses additional conventions beyond those specified in the CSTS Specification Framework, the ‘Conventions’ subsection shall include the statement specified in (4.3.7), augmented by the description of the additional conventions.

4.3.9 The ‘References’ subsection shall include:

a) the version of the CSTS Specification Framework Recommended Standard that is in effect when the CSTS Specification becomes a Recommended Standard;

b) the version of the CSTS Guidelines Recommended Practice that is in effect when the CSTS Specification becomes a Recommended Standard;

c) any other document that defines terms, concepts, processes, algorithms, etc., that are explicitly used in the CSTS specification; for a service derived from an existing CSTS, this must include the parent CSTS.

4.4 ‘OVERVIEW OF THE <NAME OF SERVICE>’ SECTION

The CSTS specification shall contain an informative section 2 entitled ‘Overview of the <name of service>’.

NOTE – Figure 4-1 is an example of the section-2 part of the table of contents for a CSTS Specification.

Section Page

….

2. OVERVIEW

2.1 SERVICE SUMMARY

2.2 FUNCTIONAL DESCRIPTION

2.2.1 SERVICE PRODUCTION

2.2.2 SERVICE PROVISION

2.3 SERVICE MANAGEMENT

2.4 CROSS SUPPORT VIEW

2.5 OPERATIONAL SCENARIO

Figure 4-1: Example Overview Section Part of the Table of Contents for a CSTS Specification

4.4.1 'SERVICE SUMMARY' SUBSECTION

The 'Overview' section shall contain a subsection entitled 'Service Summary', which shall provide a summary of the technical capabilities of the service.

4.4.2 'FUNCTIONAL DESCRIPTION' SUBSECTION

4.4.2.1 General

The 'Overview' section shall contain a subsection entitled 'Functional Description', which shall describe the functional division between the *production* of the CSTS and the *provision* of the CSTS.

4.4.2.2 'Service Production' Subsection

4.4.2.2.1 The 'Functional Description' subsection shall contain a subsection entitled 'Service Production', which shall provide a high-level description of the functions performed by the production processes associated with the CSTS.

4.4.2.2.2 If all or part of the production is performed in accordance with specific standards, the subsection shall identify those standards by reference.

4.4.2.2.3 If any production process(es) is(are) service-unique and not specified in existing standards, the subsection shall identify the annex of the CSTS specification in which that(those) production process(es) is(are) defined.

4.4.2.3 ‘Service Provision’ Subsection

The ‘Functional Description’ subsection shall contain a subsection entitled ‘Service Provision’, which shall provide a high-level description of the functions that comprise the CSTS, in terms of the application domain that is addressed by the CSTS.

NOTE – It is not the purpose of this subsection to identify the CSTS Specification Framework procedures and operations that comprise the service. Rather, it is to describe what the service does in ‘user-oriented’ terms. For example: ‘The Monitored Data CSTS allows the user to subscribe to one or more of the monitored data parameters that are published by that the service-providing Complex’.

4.4.3 ‘SERVICE MANAGEMENT’ SUBSECTION

The ‘Overview’ section shall contain a subsection entitled ‘Service Management’, which shall summarize the relationship between the CSTS and Service Management. At a high level, it shall identify what aspects of CSTS production and provision are subject to Service Management and describe the sequence of interactions between the service user and service provider that lead to the configuration and establishment of the CSTS.

4.4.4 ‘CROSS SUPPORT VIEW’ SUBSECTION

The ‘Overview’ section shall contain a subsection entitled ‘Cross Support View’, which shall describe the CSTS in the context of its role in cross support. It places the CSTS and any functions associated with the production of the CSTS in the context of the Cross Support Complex that provides the service to the service user.

4.4.5 ‘OPERATIONAL SCENARIO’ SUBSECTION

The ‘Overview’ section shall contain a subsection entitled ‘Operational Scenario’, which shall describe the lifecycle of a CSTS instance, from the establishment of the capability to provide the service via Service Management, through the scheduling of the CSTS, followed by its binding, execution, and ultimate unbinding. In the execution-phase portion of the scenario, significant capabilities of the CSTS are exercised.

4.5 ‘COMPOSITION OF THE <NAME OF SERVICE>’ SECTION

4.5.1 GENERAL

The CSTS specification shall contain a normative section 3 entitled ‘Composition of the <name of service>’.

4.5.2 ‘DISCUSSION’ SUBSECTION

4.5.2.1 The ‘Composition of the <name of service>’ section shall include an informative subsection entitled ‘Discussion’.

4.5.2.2 The ‘Discussion’ subsection shall contain a statement of the form ‘The service-level Object Identifiers for the <name of service> are specified in <annex containing the service-level Object Identifiers>’.

4.5.2.3 If any of the operations or procedures of the CSTS are abstract and require further definition in order for derived services to be complete, the ‘General’ subsection shall contain a statement of the form ‘The <name of service> service is abstract, and itself cannot be implemented. Any concrete service that is derived from this abstract service must resolve all abstract elements in this service’.

4.5.3 COMPONENT PROCEDURES SUBSECTION

4.5.3.1 The ‘Composition of the <name of service>’ section shall contain a subsection entitled ‘Procedures of the <name of service>’.

4.5.3.2 The ‘Procedures of the <name of service>’ subsection shall contain a table entitled ‘Procedures of the <name of service>’ that identifies the procedure types that comprise the CSTS, with each procedure type name heading a column of the table.

4.5.3.3 If the procedure is directly adopted or refined-only from the CSTS Specification Framework or (for a derived service) the parent CSTS, the name of the procedure shall be the same as that of the parent procedure.

4.5.3.4 If the procedure is refined-and-extended, extended-only, or service-original, the name of the procedure shall be different from that of the parent procedure.

4.5.3.5 The name of the procedure that serves as the prime procedure shall be marked with ‘[P]’ following the name of the procedure in the table.

NOTE – Table 4-1 is an example of the CSTS composition table for a hypothetical CSTS that is composed of the Association Control, Cyclic Report, and Notification procedures, and that uses the Cyclic Report procedure type for its prime procedure instance.

Table 4-1: Example of a Service Composition Table for a CSTS Comprising All Directly Adopted Procedures

Procedure	Association Control	Cyclic Report [P]	Notification
Version	-	-	-
Number of Instances	1..1	1..*	0..1
Specification Approach	adopted	adopted	adopted
Source	Subsection 4.3 of reference [1]: Association Control	Subsection 4.10 of reference [1]: Cyclic Report	Subsection 4.11 of reference [1]: Notification

4.5.3.6 For each procedure, the table shall specify:

a) a Version row that specifies the version number of the procedure type. If the procedure is directly adopted from the source specification of the procedure (see c) and d) below), the entry in this row shall be '-'; if the procedure is derived, the procedure number shall be the current version of the derived procedure;

NOTE – For the example service in table 4-1, the version numbers are all '-', indicating that all of the procedures are directly adopted from their parent procedure specification.

b) a Number of Instances row that specifies the minimum number of instances that must be instantiated, and the maximum number of instances of the procedure that may be instantiated; the syntax for specifying the minimum and maximum values shall be '<min_value> .. <max_value>':

   1) if the procedure type is used for the prime procedure instance only, the Number of Instances shall be designated '1..1' to indicate that one and only one instance of the procedure is to be supported;

   2) if the procedure type is used for the prime procedure instance and for one or more mandatory secondary procedure instances:

      i) the <min_value> component of the Number of Instances shall be equal to the sum of one (for the prime procedure instance) plus one for each of mandatory secondary procedure instance;

      ii) the <max_value> component of the Number of Instances must be equal to or greater than the <min_value>; the character '*' shall be used to indicate that the service does not put an upper bound on the number of secondary instances of the procedure;

3) if the procedure type is used for secondary procedure instances only:

   i) the <min_value> component of the Number of Instances shall be equal to or greater than zero, with zero indicating that the secondary procedure is optional;

   ii) the <max_value> component of the Number of Instances must be equal to or greater than the greater of (1, <min_value>):

      a) at least one instance must always be permitted, and the service must always support at least the mandatory number of instances;

      b) the character '*' shall be used to indicate that the service specification does not put an upper bound on the number of secondary instances of the procedure and that the number of secondary instances is fixed as part of the CSTS instantiation (see 4.7);

4) for the Association Control procedure, the designation shall be '1..1' to indicate that one and only one instance is both mandatory and permitted;

NOTES

1 Some examples are:

– for a procedure type designated as being used for the prime procedure instance, '1..*' indicates that one prime instance of the procedure is mandatory and one or more secondary instances of that procedure are permitted;

– for a procedure type designated for use as secondary procedure instances only, '1..*' indicates that one instance of the procedure is mandatory and additional instances of that secondary procedure are permitted;

– for a procedure type designated as being used for the prime procedure instance, '2..*' indicates that one prime instance of the procedure is mandatory, one secondary instance of the procedure is mandatory, and multiple additional instances are permitted;

– for a procedure type designated as being used for the prime procedure instance, '2..2' indicates that one prime instance of the procedure is mandatory, and one and only one secondary procedure instance is both mandatory and permitted;

– for a procedure type designated for use as secondary procedure instances only, '2..5' indicates that two instances of the procedure are mandatory, and up to 3 additional instances are permitted.

2 For the example service in table 4-1, '1..1' indicates one and only one instance of the Association Control procedure; '1..*' indicates one primary instance and unbounded possible secondary instances of the Cyclic Report procedure; and '0..1' indicates at most one optional instance of the Notification procedure.

c) a Specification Approach row that specifies whether the procedure is directly adopted ('adopted'), a refinement of the parent procedure ('refined-only'), an extension of the parent procedure ('extended-only'), both an extension and a refinement ('refined-and-extended'), or specifically created for the service ('service-original');

NOTE – For the example service in table 4-1, the Specification Approach for all three procedure types is 'adopted', indicating that they are directly from the parent procedure specification.

d) a Source Row that specifies the document that is the basis for this procedure. The source document must be listed in the 'References' subsection of the CSTS specification;

e) if the procedure is directly adopted, refined and/or extended from a procedure specified in the source document, the parent procedure shall also be identified in the Source Row.

NOTES

1 For the example service in table 4-1, the Source for all three procedure types is the CSTS Specification Framework, which (for this example) is listed as reference [1] in the service specification's 'References' subsection.

2 As specified in 4.5.3.3, a refined-only procedure has the same name as that of its parent procedure. If a service simply refines a procedure (for example, by refining the value of an operation parameter) and a service derived from that service also simply refines the same procedure, the name of the procedure in the child service will be the same as that of the parent procedure for the parent procedure. However, because the source identifies the *service* as well as the *procedure* from which the child procedure is refined, there is no ambiguity about which version of the procedure the child procedure is based upon.

4.5.3.7 If any of the procedures of the service are directly adopted from their parent procedures, the '<name of service> Procedures' subsection shall contain a statement of the form 'The <name of service> uses the <list of names of the directly adopted procedures identified in the service composition table> as defined in their source specifications, without derivation'.

NOTE – Inclusion of this statement indicates that the specification does not contain a 'Derived and Service-Original Procedures' subsection, as defined in 4.6.

4.5.3.8 If any of the procedures of the service are refined from their parent procedures, the '<name of service> Procedures' subsection shall contain for each such procedure a statement of the form 'The <name of service> uses the <list of names of the refined procedures identified in the service composition table>, which are refined from their parent procedure types'.

4.5.3.9 If any of the procedures of the service are extended from their parent procedures, the '<name of service> Procedures' subsection shall contain for each such procedure a statement of the form 'The <name of service> uses the <name of the extended procedure identified in the service composition table>, which is extended from its parent procedure type'.

4.5.3.10 If any of the procedures of the service are both refined and extended from their parent procedures, the '<name of service> Procedures' subsection shall contain for each such procedure a statement of the form 'The <name of service> uses the <name of the refined and extended procedure identified in the service composition table>, which is both refined and extended from its parent procedure type'.

4.5.3.11 If any of the procedures of the service are defined specifically for the service, the '<name of service> Procedures' subsection shall contain for each such procedure a statement of the form 'The <name of service> uses the <name of the service-original procedure identified in the service composition table> as defined in this service specification'.

4.5.3.12 If there are no new configuration parameters needed to configure the CSTS or otherwise support the operation of the CSTS beyond those already defined for the procedures that are adopted or derived by the CSTS, the 'Composition of the <name of service>' section shall contain a statement of the form 'The configuration information associated with the <name of service> consists solely of the configuration parameters specified for the procedures that are adopted or derived for the service. There is no additional configuration information defined for the service.'

NOTE – Inclusion of this statement indicates that the '<Name of Service> Production' annex, as defined in 4.10, does not specify additional service-type-specific configuration parameters.

4.5.3.13 If there are no new parameters needed to be monitored by the CSTS beyond those already defined for the procedures that are adopted or derived by the CSTS, the 'Composition of the <name of service>' section shall contain a statement of the form 'The monitored information associated with the <name of service> consists solely of the monitored parameters specified for the procedures that are adopted or derived for the service. There is no additional monitored information defined for the service'.

NOTE – Inclusion of this statement indicates that the '<Name of Service> Production' annex, as defined in 4.10, does not specify additional monitored parameters that need to be evaluated to determine the production status.

4.5.3.14 A CSTS shall as a minimum include the Association Control procedure and a prime procedure in the table entitled ‘<name of service> Procedures’.

4.5.3.15 Discussion: Example of a CSTS Composition Table Containing an Extended Procedure

Table 4-2 is an example of the CSTS composition table for a hypothetical CSTS that uses the Cyclic Report procedure as the procedure type of its prime procedure instance, allows an unbounded number of secondary instances of the (directly adopted) Cyclic Report procedure, allows an unbounded number of non-mandatory secondary instances of the Temperatures Cyclic Report procedure (which is extended from Cyclic Report), and allows at most one non-mandatory instance of the (directly adopted) Notification procedure.

Table 4-2: Example of a Service Composition Table for a CSTS Containing a Derived Procedure

Procedure	Association Control	Cyclic Report [P]	Temperatures Cyclic Report	Notification
Version	-	-	1	-
Number of Instances	1..1	1..*	0..*	0..1
Specification Approach	adopted	adopted	extended	adopted
Source	Subsection 4.3 of reference [1]: Association Control	Subsection 4.10 of reference [1]: Cyclic Report	Subsection 4.10 of reference [1]: Cyclic Report	Subsection 4.11 of reference [1]: Notification

4.5.3.16 Discussion: Example of CSTS Composition Table Containing a Service-Original Procedure

Table 4-3 is an example of the CSTS composition table for a hypothetical CSTS that uses the Cyclic Report procedure as the procedure type of its prime procedure instance, allows an unbounded number of secondary instances of the (directly adopted) Cyclic Report procedure, allows an unbounded number of non-mandatory secondary instances of a service-original Expedited Delivery procedure, and allows at most one non-mandatory instance of the (directly adopted) Notification procedure.

Table 4-3: Example of a Service Composition Table for a CSTS Containing a Service-Original Procedure

Procedure	Association Control	Cyclic Report [P]	Expedited Delivery	Notification
Version	-	-	1	-
Number of Instances	1..1	1..*	0..*	0..1
Specification Approach	adopted	adopted	service-original	adopted
Source	Subsection 4.3 of reference [1]: Association Control	Subsection 4.10 of reference [1]: Cyclic Report	This CSTS Specification Recommended Standard: Expedited Delivery	Subsection 4.11 of reference [1]: Notification

4.5.4 '<NAME OF SERVICE> STATE MACHINE' SUBSECTION

4.5.4.1 The 'Composition of the <name of service>' section shall contain a subsection entitled '<name of service> State Machine'.

4.5.4.2 If the CSTS is derived from an existing CSTS and behaves in accordance with the state machine of that parent CSTS, the '<name of service> State Machine' subsection shall consist of a statement of the form 'The <name of service> implements the state table of the <name of parent CSTS> [<reference number of parent CSTS>]'.

4.5.4.3 If the CSTS behaves in accordance with one of the standard CSTS state machine types specified in annex G of the CSTS Specification Framework (reference [1]), the '<name of service> State Machine' subsection shall consist of a statement of the form 'The <name of service> state machine shall conform to the state machine for a <CSTS state machine type>, as defined in the <Service State Tables annex of the CSTS Specification Framework>, reference [<Framework reference>]', where <CSTS state machine type> is one of the following:

a) 'CSTS with a stateless prime procedure instance';

b) 'CSTS with a stateful prime procedure instance'.

4.5.4.4 If the procedures that comprise the CSTS do not collectively behave in accordance with the state machine of the parent CSTS (for a derived CSTS) or one of the standard CSTS state machine types as defined in annex G of reference [1], the ‘<name of service> State Machine’ subsection shall contain:

a) a table entitled ‘<name of service> State Table’;

b) a table entitled ‘<name of service> Event Description References’;

c) if predicates are used in the state table, a table entitled ‘<name of service> Predicate Description’; and

d) if compound actions are used in the state table, a table entitled ‘<name of service> Compound Action Definitions’.

4.6 REFINED, EXTENDED, AND SERVICE-ORIGINAL PROCEDURES SECTIONS

4.6.1 If any of the procedures in the service composition table in the ‘Composition of the <name of service>’ section are refined and/or extended from another procedure or defined originally for the CSTS, the CSTS shall contain a normative section for each such procedure that is named for that procedure and defines that procedure.

NOTE – In the remainder of this Recommended Practice, each such section is referred to as a ‘Procedures’ section.

4.6.2 ‘PROCEDURES’ SECTION SUBSECTIONS

4.6.2.1 The subsection for each refined-only procedure shall contain ‘Discussion’, ‘Procedure Type Identifier’, ‘Refinement’, ‘Behavior’, ‘Required Operations’, ‘Configuration Parameters’, and ‘Procedure State Table’ subsections.

4.6.2.2 The subsection for each extended-only procedure shall contain ‘Discussion’, ‘Procedure Type Identifier’, ‘Extension’, ‘Behavior’, ‘Required Operations’, ‘Configuration Parameters’, and ‘Procedure State Table’ subsections.

4.6.2.3 The subsection for each refined-and-extended procedure shall contain ‘Discussion’, ‘Procedure Type Identifier’, ‘Refinement’, ‘Extension’, ‘Behavior’, ‘Required Operations’, ‘Configuration Parameters’, and ‘Procedure State Table’ subsections.

4.6.2.4 The subsection for each service-original procedure shall contain ‘Discussion’, ‘Procedure Type Identifier’, ‘Behavior’, ‘Required Operations’, ‘Configuration Parameters’, and ‘Procedure State Table’ subsections.

4.6.3 ‘DISCUSSION’ SUBSECTION

The ‘Discussion’ subsection shall contain informative ‘Purpose’ and ‘Concepts’ subsections.

4.6.3.1 ‘Purpose’ Subsection

The ‘Purpose’ subsection shall describe the purpose of the procedure. If the procedure is derived from another procedure, the description of the purpose should be sufficient to differentiate the procedure from the procedure from which it is derived.

4.6.3.2 ‘Concept’ Subsection

The ‘Concept’ subsection shall describe the concept of the procedure.

4.6.4 ‘PROCEDURE TYPE IDENTIFIER’ SUBSECTION

4.6.4.1 If the procedure is refined-only, the ‘Procedure Type Identifier’ subsection shall contain a requirement of the form: ‘The type identifier for the <name of CSTS> <name of procedure> shall be the same as the <name of parent procedure>’.

NOTES

1 The name of a refined-only procedure does not change from that of the parent procedure.

2 For example:

> The procedure type identifier for the Startracker Service Cyclic Report procedure shall be the same as the CSTS Specification Framework Cyclic Report procedure.

4.6.4.2 If the procedure is extended-only, refined-and-extended, or service-original, the ‘Procedure Type Identifier’ subsection shall contain a statement of the form ‘The procedure type identifier <`name of procedure identifie`r>, as specified in <Service OIDs annex reference>, shall be used for this procedure’ where <`name of procedure identifier`> is the name of the OID that is defined for the procedure in the Service OID module, and <Service OIDs annex reference> is the annex that contains the Service OIDs module.

NOTE – For example:

> The procedure type identifier `StartrackerBufferedDataDelivery`, as specified in annex xx, shall be used for this procedure.

4.6.5 ‘REFINEMENT’ SUBSECTION

The ‘Refinement’ subsection shall summarize the refined parameter values of operations of the parent procedure.

NOTE – The ‘Refinement’ subsection appears only in procedures specifications for refined-only and refined-and-extended procedures.

4.6.6 ‘EXTENSION’ SUBSECTION

The ‘Extension’ subsection shall summarize whether the derived procedure extends the parent procedure through modification of behavior, addition of operations, addition of parameters, and/or addition of parameter values.

NOTES

1 The ‘Extension’ subsection appears only in procedures specifications for extended and refined-and-extended procedures.

2 An example of the summary of the modification of the parent procedure might be

> The YYY procedure extends the XXX procedure by modification of the behavior of the procedure, addition of parameters to the NNN and MMM operations of the procedure, and the addition of the ZZZ operation.

4.6.7 ‘BEHAVIOR’ SUBSECTION

4.6.7.1 If the procedure being specified is an extended procedure and the behavior of the extended procedure is identical to that of its parent procedure, then the ‘Behavior’ subsection shall consist of the statement ‘The behavior of the <extended procedure> is the same as that of the <parent procedure>’.

4.6.7.2 If the behavior of the extended procedure is different from that of its parent procedure, or if the procedure is service-original, then the ‘Behavior’ subsection shall identify the activities of the procedure, in terms of the operations that comprise the procedure.

4.6.7.3 The scope and content of the subsections of the ‘Behavior’ subsection should be comparable to those of the procedure ‘Behavior’ subsections in the CSTS Specification Framework (reference [1]).

4.6.7.4 For each activity, the prerequisites for the activity (including but not limited to the state of the procedure) shall be identified.

4.6.7.5 For each activity involving a user-invoked operation, the behavior of the CSTS in response to the invocation shall be specified.

4.6.7.6 If the procedure is an extended procedure and the behavior is an extension of the behavior of the parent procedure’s activity, the extension shall be so identified.

NOTE – If the activity with respect to a particular operation is an extension of the activity of the parent procedure, it is sufficient to reference the activity of the parent procedure and identify only the changes that represent the extension.

4.6.8 'REQUIRED OPERATIONS' SUBSECTION

4.6.8.1 The 'Required Operations' subsection shall contain a table entitled '<procedure name> Required Operations' that identifies the operations that comprise the procedure. For each operation, the table shall specify:

a) an Extended column that specifies whether the operation is extended ('Y' or 'N') with respect to the operation in the parent procedure (for a derived procedure) or the base operation as defined in the CSTS Specification Framework (for a service-original procedure);

b) a Refined column that specifies whether the operation is refined ('Y' or 'N') with respect to the operation in the parent procedure (for a derived procedure) or the base operation as defined in the CSTS Specification Framework (for a service-original procedure);

c) a Procedure Blocking/Non-Blocking column that specifies whether the operation is blocking ('Blocking') or non-blocking ('Non-Blocking') for the procedure.

NOTE – Table 4-4 is an example of the Required Operations Table for the hypothetical Temperatures Cyclic Report procedure (see table 4-2), which extends the START operation with respect to its definition for the parent Cyclic Report procedure.

Table 4-4: Example of a Required Operations Table for a Hypothetical Temperatures Cyclic Report Derived Procedure

Operations	Extended	Refined	Procedure Blocking/Non-Blocking
START	Y	N	Blocking
STOP	N	N	Blocking
TRANSFER-DATA	N	N	Non-Blocking

NOTE – Table 4-5 is an example of the Required Operations Table for the hypothetical Expedited Delivery procedure (see table 4-3), which extends the START operation and refines the TRANSFER-DATA operation with respect to their definitions in the CSTS Specification Framework (reference [1]).

Table 4-5: Example of a Required Operations Table for a Hypothetical Expedited Delivery Service-Original Procedure

Operations	Extended	Refined	Procedure Blocking/Non-Blocking
START	Y	N	Blocking
STOP	N	N	Blocking
TRANSFER-DATA	N	Y	Non-Blocking
NOTIFY	N	N	Non-Blocking

4.6.8.2 For each extended or refined operation in the Required Operations Table, the 'Required Operations' subsection shall contain a subsection that is named for that operation and defines the extensions and/or refinements to that operation.

4.6.8.3 'Invocation[,[Acknowledgement,] Return,] and Parameters' Subsection

4.6.8.3.1 If the operation is extended or refined from an unconfirmed operation, the subsection for the operation shall contain an 'Invocation and Parameters' subsection.

4.6.8.3.2 If the operation is extended or refined from a two-phase operation, the subsection for the operation shall contain an 'Invocation, Return, and Parameters' subsection.

4.6.8.3.3 If the operation is extended or refined from a three-phase operation, the subsection for the operation shall contain an 'Invocation, Acknowledgement, Return, and Parameters' subsection.

4.6.8.3.4 Addition of Extension Parameters

4.6.8.3.4.1 If the operation is extended to include additional parameters, the 'Invocation[, [Acknowledgement,] Return,] and Parameters' subsection shall include:

a) a statement of the following form: 'In addition to the parameters of the <name of operation> invocation[, acknowledgement,] [and return] for the <name of parent procedure> as defined in [<reference number of the specification of the parent procedure>], the extension parameter(s) specified in table <table number> shall be present in the <name of operation> invocation[, acknowledgement,] [and return] of the <name of derived procedure> procedure';

b) a normative table entitled '<operation name> Extension Parameters' that names every extension parameter and indicates whether it is mandatory, conditional, or absent from the invocation, acknowledgement, and/or return (as appropriate to the operation).

NOTE – The example of the extension parameters table for the START operation of the hypothetical Temperatures Cyclic Report (see table 4-4) adds the `mode-select` parameter as a mandatory parameter of the invocation:

In addition to the parameters of the START operation of the Cyclic Report procedure as defined in 4.10 of reference [1], the extension parameter specified in table 4-6 shall be present in the START invocation of the Temperatures Cyclic Report procedure.

Table 4-6: START Extension Parameter

Extension Parameters	Invocation	Return
mode-select	M	

4.6.8.3.4.2 If the operation is extended to include additional parameters, the ‘Invocation[, [Acknowledgement,] Return,] and Parameters’ subsection shall include a subsection entitled ‘extension parameter syntax(es)’ that specifies the extension type(s) that define(s) the extension parameter syntax(es) for that operation.

NOTES

1 The definitions of the extension types are stated in the Procedure PDU module for the procedure to which the operation belongs (see 4.11).

2 For example:

The type TempCyclRptStartPosReturnExt, as defined in annex <xx>, shall define the syntax and the transfer syntax of the extension parameter of the START positive return.

3 Depending on whether the operation is unconfirmed, two-phase, or three-phase, an operation can have up to five extension types for extension parameters, one for each PDU type: invocation, positive return, negative return, positive acknowledgement, and negative acknowledgement.

4.6.8.3.4.3 If the addition of more parameters for any PDU type is to be prohibited in any child services that may be derived from the CSTS, the ‘Invocation[, [Acknowledgement,] Return,] and Parameters’ subsection shall contain a normative statement to that effect.

NOTE – For example:

> Services derived from the MyNewService CSTS shall have no additional parameters for the START positive return for any procedure derived from the Temperatures Cyclic Report procedure.

4.6.8.3.4.4 For each extension parameter named in the extension parameters table, the 'Invocation[, [Acknowledgement,] Return,] and Parameters' subsection shall include a subsection that is named for and defines that extension parameter including, but not limited to, the units and range of values of the parameter.

4.6.8.3.4.5 For each conditional extension parameter named in the extension parameters table, the subsection for that parameter shall specify the conditions under which the parameter appears in the invocation, acknowledgement, and/or return.

4.6.8.3.5 '<parameter name> Parameter Refinement' Subsections

4.6.8.3.5.1 For each refined parameter of the operation (see 3.11.2), the 'Invocation[, [Acknowledgement,] Return,] and Parameters' subsection shall include a subsection entitled '<parameter name> Parameter Refinement' that specifies the refinement of that parameter for the given procedure.

4.6.8.3.5.2 The subsection shall specify the refinement of the parameters for the given procedure.

NOTE – For example:

> The `data` parameter shall be formatted as a CCSDS Version 1 Transfer Frame, as defined in <Transfer Frame format specification>.

4.6.8.3.6 Abstract Parameters Subsections

4.6.8.3.6.1 For each abstract parameter that the service resolves, the 'Invocation[, [Acknowledgement,] Return,] and Parameters' subsection shall include a subsection entitled '<parameter name> Parameter Resolution' that specifies the resolution of the parameter for the procedure.

4.6.8.3.6.2 If the extension of the resolved parameter is to be prohibited in any child services that may be derived from the CSTS, the '<parameter name> Parameter Resolution' subsection shall contain the following sentence: 'Further extension of the <parameter name> parameter for this operation of this procedure is prohibited in services derived from this service'.

NOTE – For example:

> The `data` parameter shall be of type `OCTET STRING`.

4.6.8.3.7 ‘Return `diagnostic`’ Subsection

4.6.8.3.7.1 If the operation has one or more `diagnostic` parameter values that are to be added to the negative return of the parent operation, the ‘Invocation,[Acknowledgement,] Return, and Parameters’ subsection shall include a ‘Return `diagnostic`’ subsection.

4.6.8.3.7.2 The ‘Return `diagnostic`’ subsection shall contain text that:

a) identifies the `diagnostic` parameter of the parent operation negative return as the source of inherited `diagnostic` values; and

b) names and defines the new `diagnostic` value(s) to be added.

NOTE – For example:

> The `diagnostic` parameter shall be present in the negative return and shall have one of the following values:
>
> – one of the `diagnostic` values specified for the START operation for the Buffered Data Delivery procedure specified in 4.5 of reference [1]; or
>
> – ‘invalid mode’—the selected mode is not valid for the current configuration.

4.6.8.3.7.3 If the addition of more diagnostic values is to be prohibited in any child services that may be derived from the CSTS, the text shall contain the following sentence: ‘Addition of `diagnostic` values for the negative return for this operation of this procedure is prohibited in services derived from this service’.

4.6.8.3.8 ‘Acknowledgement `diagnostic`’ Subsection

4.6.8.3.8.1 If the operation has one or more `diagnostic` parameter values that are to be added to the negative acknowledgement of the parent operation, the ‘Invocation, Acknowledgement, Return, and Parameters’ subsection shall include an ‘Acknowledgement `diagnostic`’ subsection.

4.6.8.3.8.2 The ‘Acknowledgement `diagnostic`’ subsection shall contain text that:

a) identifies the `diagnostic` parameter of the parent operation negative acknowledgement as the source of inherited `diagnostic` values;

b) names and defines the new `diagnostic` value(s) to be added.

4.6.8.3.8.3 If the addition of more acknowledgement `diagnostic` values is to be prohibited in any child services that may be derived from the CSTS, the text shall contain the following sentence: ‘Addition of `diagnostic` values for the negative acknowledgement for this operation of this procedure is prohibited in services derived from this service’.

4.6.8.3.9 ‘`event-name` Extension’ Subsection

4.6.8.3.9.1 If the procedure includes a NOTIFY operation that (a) adds one or more notifiable events to those of the parent operation and/or (b) adds content to the one or more of the published events by specifying a to be used `event-value` parameter, the ‘Invocation[, [Acknowledgement,] Return,] and Parameters’ subsection shall include an ‘`event-name` Extension’ subsection.

NOTE – According to the CSTS Specification Framework (reference [1]) a notifiable event is composed of an `event-name` (in form of (a) a Functional Resource Name or a procedure instance identifier and (b) an Event Identifier) and an `event-value`. The Event Identifier argument is also referred to as ‘published event’.

4.6.8.3.9.2 If the NOTIFY operation adds notifiable events to the `event-name` parameter, the ‘`event-name` Extension’ subsection shall contain text that:

a) identifies for each notifiable event the `event-name` and the associated `event-value` of the parent NOTIFY operation invocation as the source of inherited notifiable events;

b) defines for each added notifiable event the new `event-name` value and the associated `event-value`.

NOTES

1 For example:

The `event-name` parameter shall be present in the invocation and its value shall be one of the following:

– one of the `event-name` values specified for the NOTIFY operation in the CSTS Specification Framework (reference [1]); or

– ‘blivet timeout’ (`event-name`)—The blivet timer has timed out and reset.

2 More examples of such extension including the specification of the associated `event-value` parameter can be found in the specification of the Buffered Data Processing and the Sequence-Controlled Data Processing procedures of the CSTS Specification Framework (reference [1]).

4.6.8.3.9.3 If the addition of notifiable events is to be prohibited in any child services that may be derived from the CSTS, the text shall contain the following sentence: ‘Addition of notifiable events for this operation of this procedure is prohibited in services derived from this service.’

4.6.8.3.9.4 If the NOTIFY operation adds content to a ‘production status change’ notifiable event, the ‘`event-value` extension’ subsection shall contain text that:

a) identifies the `event-name` parameter of the parent NOTIFY operation invocation as the source of inherited notifiable event definitions;

b) names and defines the new `event-value` content to be added where such extension can be accomplished by using a new version of the `svcProductionStatus` parameter.

NOTE – For example:

The ‘production status change’ published event of the `event-name` parameter shall be accompanied by an enhanced `event-value` parameter, which contains not only the `production-status` value ‘interrupted’, but in addition an implementation-specific integer value identifying the type of fault that caused the interruption. If this fault code for a specific interruption is undefined or if the implementation does not support fault codes, the value shall be set such that the unavailability of such fault code is flagged.

4.6.8.3.9.5 If the addition of production status content is to be prohibited in any child services that may be derived from the CSTS, the text shall contain the following sentence: ‘Addition of content to the ‘production status change’ notification is prohibited in services derived from this service.’

4.6.8.3.10 ‘`qualified-parameters` Extension’ Subsection

4.6.8.3.10.1 If the procedure includes a GET operation or a Cyclic Report-derived procedure that requires one or more additional types for valid values of the `qualified-parameters` parameter in addition to those of the parent operation, the ‘Invocation, [Acknowledgement,] Return, and Parameters’ subsection shall include a ‘`qualified-parameters` Extension’ subsection.

4.6.8.3.10.2 In accordance with annex C of the CSTS Specification Framework (reference [1]), the ‘`qualified-parameters` Extension’ subsection shall contain text that:

a) identifies the `qualified-parameters` parameter of the parent procedure as the source of inherited `qualified-parameters` value types; and

b) names and defines the new `qualified-parameters` value type(s) to be added.

NOTE – For example:

The `qualified-parameters` parameter shall have a value of one of the following types:

– one of the `qualified-parameters` valid value types specified for the common GET operation of the CSTS Specification Framework (reference [1]); or

– ‘complex’—a pair of numbers, the first representing the real component and the second representing the imaginary component.

4.6.8.3.10.3 If the addition of more valid value types is to be prohibited in any child services that may be derived from the CSTS, the text shall contain the following sentence: ‘Addition of value types for this parameter of this operation for this procedure is prohibited in services derived from this service.’

4.6.9 ‘CONFIGURATION PARAMETERS’ SUBSECTION

4.6.9.1 The ‘Configuration Parameters’ subsection shall contain the specification of the parameters required to be configured for that procedure.

4.6.9.2 If the derived service is concrete and the procedure is adopted,

a) there is no subsection describing that procedure and there cannot be a ‘Configuration Parameters’ subsection for that procedure either;

b) the ‘Setting of Service Management and Configuration Parameters Inherited from CSTS Specification Framework Procedures’ subsection (refer to 4.7) shall contain a subsection that addresses each of the configuration parameters of the adopted procedure and specifies how their values are set.

4.6.9.3 If the derived service is concrete and the procedure is derived or service-original, then

a) the section describing the derived/service-original procedure shall contain a ‘Configuration Parameters’ subsection for that procedure:

   1) if the procedure adds no new configuration parameters, the ‘Configuration Parameters’ subsection shall contain a simple statement to that effect;

   2) if the procedure adds one or more new configuration parameters, the ‘Configuration Parameters’ subsection shall contain a table similar to the <procedure type> Procedure Configuration Parameters table for the CSTS Specification Framework procedures;

b) the ‘Setting of Service Management and Configuration Parameters Inherited from CSTS Specification Framework Procedures’ subsection (refer to 4.7) shall contain a subsection that addresses each of the configuration parameters of each derived or service-original procedure and specifies how its value is set.

4.6.9.4 If the derived service is abstract and the procedure is adopted,

a) there is no section describing that procedure and there cannot be a ‘Configuration Parameters’ subsection for that procedure either;

b) the ‘Setting of Service Management and Configuration Parameters Inherited from CSTS Specification Framework Procedures’ subsection (refer to 4.7) shall contain a subsection that addresses each of the configuration parameters of the adopted procedure; because the service is abstract, the subsection may define how it is set (if it is known that all derived concrete service will use that method) or it may defer the method to the derived concrete services.

4.6.9.5 If the derived service is abstract and the procedure is derived or service-original the section describing the derived or service-original procedure shall contain a ‘Configuration Parameters’ subsection for that procedure:

a) if the procedure adds no new configuration parameters, the ‘Configuration Parameters’ subsection shall contain a simple statement to that effect;

b) if the procedure adds one or more new configuration parameters, the ‘Configuration Parameters’ subsection shall contain a table similar to the <procedure type> Procedure Configuration Parameters table for the CSTS Specification Framework procedures specified in reference [1] that lists the new configuration parameters;

c) For each procedure adding new configuration parameters the subsection ‘Setting of Configuration Parameters’ in an abstract service specification may identify the method by which the parameter is set or may defer the specification of the method to the concrete service derived from the abstract service.

NOTE – In a concrete service specification derived from an abstract service, the method by which each of the configuration parameters is set needs to be specified.

4.6.10 ‘PROCEDURE STATE TABLES’ SUBSECTION

4.6.10.1 If the procedure is a derived procedure and the behavior of the derived procedure is the same as the state table of the parent procedure, then the ‘Procedure State Tables’ subsection shall consist of the statement ‘The <name of derived procedure> procedure adopts the state table of the <name of parent procedure> procedure without modification.’

4.6.10.2 If the behavior of the derived procedure differs from the state table of the parent procedure or if the procedure is a service-original procedure, then the ‘Procedure State Tables’ subsection shall contain:

a) a table entitled ‘<name of procedure> State Table’;

b) a table entitled ‘Event Description References’;

c) if predicates are used in the state table, a table entitled ‘Predicate Description’;

d) if simple actions are used in the state table, a table entitled ‘Simple Action References’; and

e) if compound actions are used in the state table, a table entitled ‘Compound Action Definitions’.

NOTE – The format of these tables can be found in the procedures definitions in the CSTS Specification Framework (reference [1]).

4.7 ‘SETTING OF SERVICE MANAGEMENT AND CONFIGURATION PARAMETERS INHERITED FROM CSTS SPECIFICATION FRAMEWORK PROCEDURES’ SECTION

4.7.1 OVERVIEW

The procedures of the CSTS Specification Framework (reference [1]) define configuration parameters for the CSTS Specification Framework procedures, but reference [1] defers the specification of the method by which each of those configuration parameters is to be set to the services using the procedures. There may be additional parameters that need to be managed although they are not procedure configuration parameters. An example of such parameter is the `responder-port-identifier` parameter (refer to 3.4.2.2.4.3 of reference [1]).

4.7.2 SPECIFICATIONS

4.7.2.1 The CSTS specification shall contain a ‘Setting of Service Management and Configuration Parameters Inherited from CSTS Specification Framework Procedures’ section to specify the methods by which the values of the CSTS Specification Framework procedure configuration parameters are to be set for the CSTS.

4.7.2.2 For each service management parameter not being a procedure configuration parameter, the ‘Setting of Service Management and Configuration Parameters Inherited from CSTS Specification Framework Procedures’ section shall contain a subsection entitled ‘<parameter name> Service Management Parameter’ specifying the classifier of the parameter.

4.7.2.3 For each procedure type in the service, the ‘Setting of Service Management and Configuration Parameters Inherited from CSTS Specification Framework Procedures’ section shall contain a subsection entitled ‘<procedure name> Procedure Configuration Parameters’.

4.7.2.4 Each '<procedure name> Procedure Configuration Parameters' subsection shall contain a set of normative statements declaring the method by which the value of each and every one of the procedure configuration parameters is to be set for that procedure.

NOTES

1 The possible methods for setting the configuration parameter values include, but are not limited to, specifying the parameter to be a service management parameter, setting the parameter to a fixed value in the CSTS specification, and deferring specification of the method to the individual implementations.

2 An example 'Association Control Procedure Configuration Parameters' subsection is as follows:

7.2 ASSOCIATION CONTROL PROCEDURE CONFIGURATION PARAMETERS

7.2.1 The service-user-responding-timer (refer to table 4-2 in 4.3.5 of reference [1]) shall be configured by a service management parameter with the classifier <name of service>ServiceUserRespondingTimer.

7.2.2 The `initiator-identifier` (refer to table 4-2 in 4.3.5 of reference [1]) shall be configured by a service management parameter with the classifier <name of service>`InitiatorId.`

7.2.3 The `responder-identifier` (refer to table 4-2 in 4.3.5 of reference [1]) shall be configured by a service management parameter with the classifier <name of service>`ResponderId`.

7.2.4 The `service-instance-identifier` (refer to table 4-2 in 4.3.5 of reference [1]) shall be configured by a service management parameter with the classifier <name of service>`ServiceInstanceId`.

4.8 <NAME OF SERVICE> SERVICE-SPECIFIC VERSIONS OF SERVICE-GENERIC PARAMETERS AND EVENTS

4.8.1 OVERVIEW

Annex B of reference [1] specifies service-generic parameters and events for use by any CSTS. To the extent a CSTS uses them, it has to specify the service-specific labels for the `production-status` parameter and the 'production status change' and 'production configuration change' events.

4.8.2 SPECIFICATION

4.8.2.1 If the CSTS uses the generic `production-status` parameter, the '<name of service> Service-Specific Versions of Service-Generic Parameters and Events' section shall contain a subsection entitled '<service name>SvcProductionStatus Parameter' specifying the label of the service-specific `production-status` parameter and the associated Object Identifier.

4.8.2.2 If the CSTS uses the generic 'production status change' event, the '<name of service> Service-Specific Versions of Service-Generic Parameters and Events' section shall contain a subsection entitled '<service name>SvcProductionStatusChange Event' specifying the label of the service-specific 'production status change' event and the associated Object Identifier.

4.8.2.3 If the CSTS uses the generic 'production configuration change' event, the '<name of service> Service-Specific Versions of Service-Generic Parameters and Events' section shall contain a subsection entitled '<service name>SvcProductionConfigurationChange Event' specifying the label of the service-specific 'production configuration change' event and the associated Object Identifier.

4.9 'REFINEMENT AND EXCLUSION OF DEFINITIONS OF CSTS SPECIFICATION FRAMEWORK PARAMETERS, EVENTS, DIRECTIVES, AND `diagnostic` VALUES USED BY THE <NAME OF SERVICE>' SECTION

4.9.1 If the CSTS refines the definition of one or more CSTS Specification Framework parameters, events, directives or `diagnostic` values or excludes any of those, the CSTS specification shall contain a normative section entitled 'Refinement and Exclusion of Definitions of CSTS Specification Framework [Parameters,] [Events,] [Directives,] [and `diagnostic` Values] Used by the <name of service>', where each of the terms in square brackets appear in the section title only if the service refines that category.

NOTE – For example, for a service that refines only some CSTS Specification Framework parameters and events, the section title would be 'Refinement of Definitions of CSTS Specification Framework Parameters and Events used by the <name of service>'.

4.9.2 If the CSTS refines the definition of one or more CSTS Specification Framework parameters, the 'Refinement and Exclusion of Definitions of CSTS Specification Framework Parameters, [Events,] [Directives,] [and `diagnostic` Values] Used by the <name of service> Service' section shall contain a subsection entitled 'Parameter Definition Refinement'.

4.9.3 For each CSTS Specification Framework parameter whose definition is refined by the CSTS, the 'Parameter Definition Refinement' subsection shall contain a normative specification of the refinement that (a) identifies the parameter that is being refined, (b) identifies the operation(s), procedure(s), and/or service-generic category to which the parameter definition refinement applies, and (c) provides the refined definition of the parameter.

NOTE – The following is an example CSTS Specification Framework parameter refinement specification:

> The definition of the `data` parameter used by the Cyclic Report procedure TRANSFER-DATA operation (see <reference to the parent procedure>) shall be refined as follows:
>
> `data` <refined definition of the parameter>.

4.9.4 If the CSTS refines the definition of one or more CSTS Specification Framework events, the ‘Refinement or Exclusion of Definitions of CSTS Specification Framework [Parameters,] Events, [Directives,] [and `diagnostic` Values] Used by the <name of service>’ section shall contain a subsection entitled ‘Event Definition Refinement’.

4.9.5 For each CSTS Specification Framework event whose definition is refined by the CSTS, the ‘Event Definition Refinement’ subsection shall contain a normative specification of the refinement that (a) identifies the event that is being refined, (b) identifies the operation(s), procedure(s), and/or service-generic category to which the event definition refinement applies, and (c) provides the refined definition of the event.

NOTE – The following is an example CSTS Specification Framework event refinement specification:

> The definition of the ‘resource configuration change’ event with the `event-value` set to ‘configured’ used by the Notification procedure NOTIFY operation (see <reference>) shall be refined as follows:
>
> ‘resource configuration change’ event with the `event-value` set to ‘configured’: Configuration of the XYZ Functional Resource Instance has been completed.

4.9.6 If the CSTS refines the definition of one or more CSTS Specification Framework directives, the ‘Refinement or Exclusion of Definitions of CSTS Specification Framework [Parameters,] [Events,] Directives, [and `diagnostic` Values] Used by the <name of service>’ section shall contain a subsection entitled ‘Directive Definition Refinement’.

4.9.7 For each CSTS Specification Framework directive whose definition is refined by the CSTS, the ‘Directive Definition Refinement’ subsection shall contain a normative specification of the refinement that (a) identifies the directive that is being refined, (b) identifies the procedure(s) to which the directive definition refinement applies, and (c) provides the refined definition of the directive.

4.9.8 If the CSTS refines the definition of one or more CSTS Specification Framework `diagnostic` values, the ‘Refinement or Exclusion of Definitions of CSTS Specification Framework [Parameters,] [Events,] [Directives,] and `diagnostic` Values Used by the <name of service>’ section shall contain a subsection entitled ‘`diagnostic` Value Definition Refinement’.

4.9.9 For each CSTS Specification Framework `diagnostic` value whose definition is refined by the CSTS, the ‘`diagnostic` Value Definition Refinement’ subsection shall contain a normative specification of the refinement that (a) identifies the `diagnostic` value that is being refined, (b) identifies the operation(s) and procedure(s) to which the `diagnostic` value definition refinement applies, and (c) provides the refined definition of the `diagnostic` value.

NOTE – The following is an example CSTS Specification Framework `diagnostic` value refinement specification:

The definition of the ‘unknown Functional Resource Type’ `diagnostic` value used by the Cyclic Report procedure START operation (see <reference>) and the Information Query procedure GET operation (see <reference>) shall be refined as follows:

‘unknown Functional Resource Type’—the Functional Resource Type contained in the `list-of-parameters` parameter is unknown to the service provider or the Functional Resource Type is not associated with any Functional Resource that is configured as part of the service package. The unknown Functional Resource Type shall be returned with the `diagnostic`.

4.9.10 If the CSTS excludes one or more CSTS Specification Framework parameters, the ‘Refinement and Exclusion of Definitions of CSTS Specification Framework Parameters, [Events,] [Directives,] [and `diagnostic` Values] Used by the <name of service> Service’ section shall contain a subsection entitled ‘Parameters Exclusion’.

4.9.11 If the CSTS excludes one or more CSTS Specification Framework events, the ‘Refinement and Exclusion of Definitions of CSTS Specification Framework [Parameters,] Events, [Directives,] [and `diagnostic` Values] Used by the <name of service> Service’ section shall contain a subsection entitled ‘Events Exclusion’.

4.9.12 If the CSTS excludes one or more CSTS Specification Framework directives, the ‘Refinement and Exclusion of Definitions of CSTS Specification Framework [Parameters,] [Events,] Directives, [and `diagnostic` Values] Used by the <name of service> Service’ section shall contain a subsection entitled ‘Directives Exclusion’.

4.9.13 If the CSTS excludes one or more CSTS Specification Framework `diagnostic` values, the ‘Refinement and Exclusion of Definitions of CSTS Specification Framework [Parameters,] [Events,] [Directives,] [and] diagnostic Values Used by the <name of service> Service’ section shall contain a subsection entitled ‘`diagnostic` Values Exclusion’.

4.10 SERVICE OBJECT IDENTIFIERS MODULE ANNEX

4.10.1 The CSTS specification shall contain a normative annex entitled ‘SERVICE OBJECT IDENTIFIERS MODULE’, hereinafter referred to as the ‘OID Module’.

4.10.2 The content of the OID Module shall conform to the specification of an ASN.1 module as specified in the Module definition section of the ASN.1 Specification of Basic Notation (reference [4]).

4.10.3 MODULE IDENTIFIER

4.10.3.1 For a CSTS that is not derived from another CSTS, the OID Module shall specify the module identifier (see reference [4]) of the module as follows:

```
CCSDS-<Module Reference Service Component>-OBJECT-IDENTIFIERS
{  iso(1) identified-organization(3) standards-producing-organization(112)
   ccsds(4) css(4) csts(1) services(2)
   <service identifier>Service(<service number>)
   <service identifier>ServiceModules(4) object-identifiers(1)
}
```

where:

a) <Module Reference Service Component> is an all-uppercase character string used to indicate the specific CSTS within the module reference (that is, the ‘CCSDS-<Module Reference Service Component>-OBJECT-IDENTIFIERS’ string);

b) <service identifier> is a character string used to name the service-specific node of the Object Identifier; and

c) <service number> is an integer that has been uniquely assigned to this CSTS and registered with SANA for this CSTS.

NOTE – The following is an example OID for the OID module for a hypothetical ‘catalog packets’ service that has <Module Reference Service Component> = ‘CATALOG-PACKETS’, <service identifier> = ‘catalogPkts’, and <service number> = 57:

```
CCSDS-CATALOG-PACKETS-OBJECT-IDENTIFIERS
{  iso(1) identified-organization(3)
   standards-producing-organization(112)
   ccsds(4) css(4) csts(1) services(2) catalogPktsService(57)
   catalogPktsServiceModules(4) object-identifiers(1)
}
```

4.10.3.2 For a CSTS that is derived from another CSTS, the OID Module shall specify the module identifier (see reference [4]) of the module as follows:

```
CCSDS-<Module Reference Service Component>-OBJECT-IDENTIFIERS
{  iso(1) identified-organization(3) standards-producing-organization(112)
   ccsds(4) css(4) csts(1) services(2)
   <parent service identifier>Service(<parent service number>)
   <parent service identifier>DerivedServices(1)
   <service identifier>Service(<service number>)
   <service identifier>ServiceModules(4) object-identifiers(1)
}
```

where:

a) <Module Reference Service Component>, <service identifier> and <service number> are as defined in 4.10.3.1;

b) <parent service identifier> is a character string used to name the service-specific node of the parent service in the Object Identifier tree;

c) <parent service number> is a number used to uniquely identify the parent service-within the Object Identifier tree.

4.10.3.3 The <Module Reference Service Component> shall be constructed such that the resulting module reference (i.e., 'CCSDS-CSTS-<Module Reference Service Component>-OBJECT-IDENTIFIERS') conforms to the format of the module reference as specified in reference [4].

NOTE – As specified in reference [4], a 'module reference' consists of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character is an uppercase letter. A hyphen is not allowed as the last character. A hyphen cannot be immediately followed by another hyphen.

4.10.3.4 All identifiers specified in 4.10.3.2 shall conform to the format of a valid identifier as specified in reference [4].

NOTE – As specified in reference [4], an 'identifier' consists of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character is a lowercase letter. A hyphen is not allowed as the last character. A hyphen cannot be immediately followed by another hyphen.

4.10.3.5 The <service number> and <parent service number> specified in 4.10.3.1 and 4.10.3.2 respectively shall conform to the format of a valid number as specified in reference [4].

NOTE – As specified in reference [4], a 'number' consists of one or more digits. The first digit is not allowed to be zero unless the 'number' is a single digit.

4.10.3.6 The <service modules subtree identifier> shall be of the form <service identifier>ServiceModules.

4.10.4 IMPORTED IDENTIFIERS

NOTE – The IMPORTS clauses discussed in this subsection assume that the OIDs associated with the newly specified CSTS type have been assigned and the ASN.1 modules and the associated registries have been updated in accordance with the applicable registry management policy.

4.10.4.1 For a CSTS that is not derived from another CSTS, the OID Module shall import the `services` identifier from the CCSDS-CSTS-OBJECT-IDENTIFIERS module defined in F4.1 of the CSTS Specification Framework (reference [1]).

NOTE – The following is an example IMPORTS clause for the OID module for a non-derived service:

```
IMPORTS services
  FROM CCSDS-CSTS-OBJECT-IDENTIFIERS
;
```

4.10.4.2 For a CSTS that is derived from another CSTS, the OID Module shall import the derived services subtree identifier from the OID module defined in the specification of the parent CSTS.

NOTE – The following is an example IMPORTS clause for the OID module for a service that is derived from a hypothetical catalog packets service:

```
IMPORTS catalogPktsDerivedServices
  FROM CCSDS-CATALOG-PACKETS-OBJECT-IDENTIFIERS
;
```

4.10.5 ROOT OBJECT IDENTIFIERS

4.10.5.1 In accordance with annex D of the CSTS Specification Framework (reference [1]), for a CSTS that is not derived from another CSTS, the OID Module shall specify the Object Identifier of the service under the services subtree (see 4.10.4.1) as:

```
<service identifier>    OBJECT IDENTIFIER ::=
   {services <service number>}
```

where <service identifier> is the identifier of the service, and <service number> is the integer value that has been uniquely assigned to this CSTS and registered with SANA for this CSTS.

4.10.5.2 In accordance with annex D of the CSTS Specification Framework (reference [1]), for a CSTS that is derived from another CSTS, the OID Module shall specify the Object Identifier of the service under the derived services subtree of the parent service (see 4.10.4.2) as:

```
<service identifier>    OBJECT IDENTIFIER ::=
   {<parent service identifier>DerivedServices <service number>}
```

where

a) <service identifier> is the identifier of the child service;

b) <parent service identifier> is the identifier of the service from which the new CSTS is derived; and

c) <service number> is the integer value that has been uniquely assigned to this CSTS and registered with SANA for this CSTS.

4.10.5.3 The values of the <service identifier> and <service number> shall be the same as the <service identifier> and <service number> in the module identifier (see 4.10.4.1 and 4.10.4.2).

NOTE – The following is an example service identifier Object Identifier assignment for a hypothetical catalog telemetry packets service that is derived from a hypothetical catalog packets service:

```
catalogTelemPkts    OBJECT IDENTIFIER ::=
  {catalogPktsDerivedServices  5}
```

4.10.5.4 In accordance with annex D of the CSTS Specification Framework (reference [1]), the OID Module shall specify the Object Identifier of the derived services subtree for the service as:

```
<derived services subtree identifier>  OBJECT IDENTIFIER ::=
   {<service identifier>  1}
```

4.10.5.5 The <derived services subtree identifier> should be of the form <service identifier>DerivedServices.

4.10.5.6 In accordance with annex D of the CSTS Specification Framework (reference [1]), the OID Module shall specify the Object Identifier of the extended service parameters subtree for the service as:

```
<extended service parameters subtree identifier>  OBJECT IDENTIFIER ::=
   {<service identifier>  2}
```

4.10.5.7 The <extended service parameters subtree identifier> should be of the form <service identifier>ExtendedServiceParameters.

4.10.5.8 In accordance with annex D of the CSTS Specification Framework (reference [1]), the OID Module shall specify the Object Identifier of the service procedures subtree for the service as:

```
<service procedures subtree identifier>  OBJECT IDENTIFIER ::=
   {<service identifier>  3}
```

4.10.5.9 The <service procedures subtree identifier> should be of the form <service identifier>ServiceProcedures.

4.10.5.10 In accordance with annex D of the CSTS Specification Framework (reference [1]), the OID Module shall specify the Object Identifier of the service modules subtree for the service as:

```
<service modules subtree identifier>  OBJECT IDENTIFIER ::=
   {<service identifier>  4}
```

4.10.5.11 The <service modules subtree identifier> should be of the form <service identifier>ServiceModules.

4.10.5.12 In accordance with annex D of the CSTS Specification Framework (reference [1]), the OID Module shall specify the Object Identifier of the service Functional Resource reference subtree for the service as:

```
<service functional resource reference subtree identifier>  OBJECT
IDENTIFIER ::=
   {<service identifier>  5}
```

4.10.5.13 The <service functional resource reference subtree identifier> should be of the form <service identifier>ServiceFrRef.

4.10.6 SERVICE PROCEDURES IDENTIFIERS

4.10.6.1 If the service has any extended procedures, the OID Module shall specify the Object Identifier of each of the extended procedures under the service procedures subtree for the service (see 4.10.5.9) as:

```
<service procedure identifier>    OBJECT IDENTIFIER ::=
   {<service procedures subtree identifier>   <service procedure number>}
```

where <service procedure identifier> is the identifier of the procedure, <service procedures subtree identifier> is the identifier of the service procedures subtree of the service, and <service procedure number> is an integer value that has been uniquely assigned to this procedure and registered with SANA for this CSTS.

4.10.6.2 The <service procedure subtree identifier> should be of the form <service identifier>SP<procedure identifier>.

4.10.6.3 For traceability and understandability, the initial part of the <service procedure identifier> should identify (possibly in an abbreviated form) the service to which the procedure belongs.

4.10.6.4 For traceability and understandability, if the procedure extends a procedure, the <service procedure identifier> should identify (possibly in an abbreviated form) the procedure from which the procedure is derived.

NOTE – The following is an example service procedure identifier Object Identifier assignment for a hypothetical Catalog Packets Notification procedure that is extended for a hypothetical catalog packets service:

```
catalogPktsNotification    OBJECT IDENTIFIER ::=
  {catalogPktsServiceSPcatalogPktsNotification  1}
```

4.10.7 DISCUSSION: EXAMPLE OBJECT IDENTIFIERS MODULE

Figure 4-2 is an example Object Identifiers module for the specification of a hypothetical Catalog Packets CSTS.

```
CCSDS-CATALOG-PACKETS-OBJECT-IDENTIFIERS
{  iso(1) identified-organization(3)
   standards-producing-organization(112) ccsds(4) css(4) csts(1)
   services(2) catalogPkts(57) catalogPktsModules(4)
   object-identifiers(1)
}

DEFINITIONS
IMPLICIT TAGS
::= BEGIN

EXPORTS
catalogPktsDerivedServices
,  catalogPktsExtServiceParameters
,  catalogPktsServiceProcedures
;

IMPORTS services
   FROM CCSDS-CSTS-OBJECT-IDENTIFIERS
;

-- ************************************************************
-- Root Object Identifiers of the Service
catalogPkts OBJECT IDENTIFIER ::= {services 57}
catalogPktsDerivedServices OBJECT IDENTIFIER ::= {catalogPkts 1}
```

4.11 PROCEDURE PDU SPECIFICATION ANNEXES

4.11.1 If the CSTS includes any refined-and-extended, extended-only, or service-original procedures, the CSTS specification shall contain a normative Procedure PDU annex for each such procedure.

4.11.2 The name of each such Procedure PDU annex shall be of the form ‘PROCEDURE <name of procedure> PDUS’.

4.11.3 PROCEDURE PDU ASN.1 MODULE

4.11.4 Each Procedure PDU annex shall contain an ASN.1 module (hereinafter referred to as a ‘Procedure PDU module’) that conforms to the specification of an ASN.1 module as specified in the Module definition section of the ASN.1 Specification of Basic Notation (reference [4]).

4.11.5 The ASN.1 module shall specify the extensions used by the service.

4.11.5.1 Module Identifier

4.11.5.1.1 For a CSTS that is not derived from another CSTS, the Procedure PDU Module shall specify the module identifier (see reference [4]) of the module as follows:

```
CCSDS-<Module Reference Service Component>-<Module Reference Procedure
Component>-PDUS

{  iso(1) identified-organization(3) standards-producing-organization(112)
   ccsds(4) css(4) csts(1) services(2)
   <service identifier>Service(<service number>)
   <service identifier>ServiceModules(4) extensions(2)
   <procedure pdu module identifier>(<procedure pdu module number>)
}
```

where <service identifier> and <service number> are as defined in 4.10.3.1 and the other components are specified as follows:

a) <Module Reference Procedure Component> is an all-uppercase character string used to indicate the specific procedure within the module reference;

b) <procedure pdu module identifier> is a character string used to identify the module as specifying the operations and extensions for the specific procedure; and

c) <procedure pdu module number> is an integer that has been uniquely assigned to this procedure for this CSTS.

4.11.5.1.2 For a CSTS that is derived from another CSTS, the Procedure PDU Module shall specify the module identifier (see reference [4]) of the module as follows:

```
CCSDS-<Module Reference Service Component>-<Module Reference Procedure
Component>-PDUS

{  iso(1) identified-organization(3) standards-producing-organization(112)
   ccsds(4) css(4) csts(1) services(2)
   <parent service identifier>Service(<parent service number>)
   <parent service identifier>DerivedServices(1)
   <service identifier>Service(<service number>)
   <service identifier>ServiceModules(4) extensions(2)
   <procedure pdu module identifier>(<procedure pdu module number>)
}
```

where:

a) <Module Reference Service Component> is defined in 4.10.3.1 a), 4.10.3.2, 4.10.3.3, <service identifier> and <service number> are defined in 4.10.3.1 b) and 4.10.3.1 c), respectively;

b) <parent service identifier> and <parent service number> are defined in 4.10.3.2; and

c) <procedure pdu module identifier>, <procedure pdu module number>, and <Module Reference Procedure Component> are defined in 4.11.5.1.1.

4.11.5.1.3 The <Module Reference Service Component> and <Module Reference Procedure Component> shall be constructed such that the resulting module reference (i.e., 'CCSDS-CSTS-<Module Reference Service Component>-<Module Reference Service Component>-PDUs') conforms to the format of the module reference as specified in reference [4].

NOTE – As specified in reference [4], a 'module reference' consists of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character is an uppercase letter. A hyphen is not allowed as the last character. A hyphen cannot be immediately followed by another hyphen.

4.11.5.1.4 The <procedure pdu module identifier> shall conform to the format of a valid identifier as specified in reference [4].

NOTE – As specified in reference [4], an 'identifier' consists of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character is a lowercase letter. A hyphen is not allowed as the last character. A hyphen cannot be immediately followed by another hyphen.

4.11.5.1.5 The <procedure pdu module number> shall conform to the format of a valid number as specified in reference [4].

NOTE – As specified in reference [4], a 'number' consists of one or more digits. The first digit is not allowed to be zero unless the 'number' is a single digit.

4.11.5.1.6 The procedure PDU module identifier should be of the form <service procedure identifier>Pdus.

4.11.5.2 Exported Identifiers and Types

4.11.5.2.1 The Procedure PDU Module shall export the procedure PDU module identifier (see 4.11.5.1.1 and 4.11.5.1.2).

4.11.5.2.2 The Procedure PDU Module shall export all extended data types that are defined within the Procedure PDU Module.

NOTES

1 Use of comments to demarcate the specification of the exported identifiers and types is encouraged.

2 In case the extended data type cannot be fully described with ASN.1, the external format has to be provided.

4.11.5.3 Imported Identifiers and Types

4.11.5.3.1 If the procedure extends a procedure of the CSTS Specification Framework or a parent service without adding operations, the Procedure PDU Module shall import the procedure PDU type of the parent procedure from the Procedure PDU Module that defines that procedure.

NOTE – The following is an example IMPORTS clause for the Procedure PDU module for a CatalogPktsNotification procedure that is derived from the CSTS Specification Framework Notification procedure without adding operations:

```
IMPORTS NotificationPdu
  FROM CCSDS-CSTS-NOTIFICATION-PDUS
;
```

4.11.5.3.2 If the procedure extends a procedure of the CSTS Specification Framework or a parent service and adds operations, or if the procedure is service-original, the Procedure PDU Module shall import the `CstsFrameworkPdu` type from the `CCSDS-CSTS-PDUS` module of the CSTS Specification Framework (F4.15 of reference [1]).

NOTE – Use of comments to demarcate the specification of the imported identifiers and types is encouraged.

4.11.6 PROCEDURE PDU TYPE SPECIFICATION

4.11.6.1 If the procedure extends a procedure of the CSTS Specification Framework or a parent service without adding operations, the Procedure PDU Module shall include a comment that (a) states that the procedure reuses the PDU defined by the procedure from which it was derived (see 4.11.5.3.1) and (b) identifies that PDU.

NOTE – The following is an example procedure PDU specification statement for a Buffered Tracking Data Message Delivery procedure that uses the same PDU as the Buffered Data Delivery procedure:

```
          -- The Buffered Tracking Data Message Delivery procedure is
          -- derived from the Buffered Data Delivery procedure. It reuses
          -- the PDU defined in the Buffered Data Delivery procedure:
          -- BufferedDataDeliveryPdu type defined in the
          -- CCSDS-CSTS-BUFFERED-DATA-DELIVERY-PDUS module of
          -- the CSTS Specification Framework, reference [1].
```

4.11.6.2 If the procedure extends a procedure of the CSTS Specification Framework or a parent service and adds operations, or if the procedure is service-original, the Procedure PDU Module shall specify the procedure PDU type as being of type CstsFrameworkPdu (see 4.11.5.3.2) with the component message types.

NOTES

1 The following is an example procedure PDU specification for a NotifiedBufferedDataDelivery procedure that is derived from the CSTS Specification Framework BufferedDataDelivery by the addition of a NOTIFY operation:

```
NotifiedBufferedDataDeliveryPdu ::= CstsFrameworkPDU (WITH COMPONENTS
{  startInvocation
,  startReturn
,  stopInvocation
,  stopReturn
,  transferBufferInvocation
,  notifyInvocation
})
```

2 Use of comments to demarcate the specification of the procedure PDU type is encouraged.

4.11.7 OPERATION EXTENSION TYPE SPECIFICATIONS

4.11.7.1 For each operation of the procedure, the Procedure PDU Module shall include a subsection that:

a) identifies the operation; and

b) includes a declaration for each extension parameter for that operation.

4.11.7.2 For each extension parameter of the operation that does not extend its source operation parameter, the declaration for that extension parameter shall:

a) state (in comments) that the operation does not add new parameters or new parameter values (as appropriate) of this extension type; and

b) identify (in comments) the parent operation extension parameter and state that it is set to ‘notUsed’.

4.11.7.3 For each extension parameter of the operation that extends its source operation parameter, the declaration for that extension parameter shall:

a) state (in comments) that the operation does add new parameters or new parameter values (as appropriate) of this extension type;

b) identify (in comments) the parent operation extension parameter;

c) specify the extension type that is to be applied to the parent operation extension parameter; and

d) specify the Object Identifier of the extension type.

NOTE – Annex B identifies the extension parameters for the base operations and the operations of the procedures of the CSTS Specification Framework (reference [1]).

4.11.7.4 Discussion—Example of PDU Module Comments for Operations

4.11.7.4.1 Non-Extended Operation Parameters

In this example, the service uses a procedure that is derived from the Buffered Data Delivery procedure defined in 4.5 of reference [1], but the START operation is not extended in any way. The START operation of the CSTS Specification Framework Buffered Data Delivery procedure has four extension parameters: new parameters for the invocation, new parameters for the positive return, new parameters for the negative return, and extension values for the `diagnostic` parameter of the negative return. The following ASN.1 comments describe that these extension parameters are to be set to 'notUsed'. The START operation of the CSTS Specification Framework Buffered Data Delivery procedure does not itself extend the invocation parameters, positive return parameters, or negative return parameters, but it does extend the `diagnostic` values of the negative return. Therefore the extension parameters that are pertinent to the derived procedure's START operation are those of the base START operation's invocation parameters, positive return parameters, and negative return parameters, and the `diagnostic` extension parameter of the Buffered Data Delivery START negative return.

```
-- *****
-- START invocation extension parameters
-- No extension parameters are added to the START Invocation.
-- The BufferedDataDelivery procedure does not extend the parameters of
-- the START invocation. Therefore, 'StartInvocation':
-- 'startInvocationExtension': 'bddStartInvocExt':
-- 'BuffDataDelStartInvocExt':
-- 'buffDataDelStartInvocExtExtension' shall be set to 'notUsed'.
-- 'startInvocationExtension': 'Extended' (see CCSDS-CSTS-COMMON-
-- OPERATIONS-PDUS module in reference [1]) is set to 'notUsed'.
-- START positive return parameters
-- No extension parameters are added to the START positive return.
-- The BufferedDataDelivery procedure does not extend the parameters of
-- the START positive return. Therefore
-- 'StartReturn': 'StandardReturnHeader': 'result': 'positive': 'Extended'
-- (see CCSDS-CSTS-COMMON-OPERATIONS-PDUS module in reference [1]) is set -
- to 'notUsed'.

-- START negative return extension parameters
-- No extension parameters are added to the START negative return.
-- The BufferedDataDelivery procedure does not extend the parameters of
-- the START negative return. Therefore
-- 'StartReturn': 'StandardReturnHeader': 'result': 'negative':
-- 'negExtension': 'Extended' (see CCSDS-CSTS-COMMON-OPERATIONS-PDUS
-- module in reference [1]) is set to 'notUsed'.

-- START negative return extension diagnostics
-- No extension diagnostics are added to the START negative return.
-- The Buffered Data Delivery procedure does not extend the diagnostic
-- values of the START negative return. Therefore the START negative
-- return makes use of: (a) one of the common diagnostics of
-- 'StandardReturnHeader': 'result': 'negative': 'diagnostic':
-- 'Diagnostic' except 'diagnosticExtension'; or (b) one of the additional
-- diagnostics defined by 'StartReturn': 'StandardReturnHeader': 'result':
-- 'negative': 'diagnostic': 'Diagnostic': 'diagnosticExtension':
-- startDiagnosticExt': 'StartDiagnosticExt' except
-- startDiagnosticExtExtension'.
-- *****
```

NOTE – Actually, the Buffered Data Delivery procedure as specified in 4.5 of reference [1] does extend the START operation diagnostics by adding permissible `diagnostic` values. The above is only an example applicable in case such extension is not defined.

4.11.7.4.2 Extended Operation Parameter

In this example, the service defines a Buffered Tracking Data Message Delivery procedure that is derived from the Buffered Data Delivery procedure of the CSTS Specification Framework, but the START positive return is extended to include a new `tdmHeader` parameter. The declarations for the START invocation new parameters, negative return new parameters, and negative return `diagnostic` values are as specified in the example in 4.11.7.4.1. The following ASN.1 declaration defines the extension type for the START positive return. Since the START operation of the CSTS Specification Framework Buffered Data Delivery procedure does not itself extend the positive return parameters, the START operation extension parameter that is extended by this new extension type is that of the base START operation's positive return.

```
-- START positive return parameters
-- START positive return is extended with the tdm-header parameter.
-- The BufferedDataDelivery procedure does not extend the parameters of
-- the START positive return. Therefore this extension applies directly to
-- the StartReturn PDU type defined in the CCSDS-CSTS-COMMON-OPERATIONS-
-- PDUS module in reference [1]:
-- 'StartReturn': 'StandardReturnHeader': 'result': 'positive':
-- 'Extended': 'external': 'Embedded': 'BuffTrkDataDelStartPosReturnExt'
--

BuffTrkDataDelStartPosReturnExt  ::=   SEQUENCE
{  tdmHeader      VisibleString
}
buffTrkDataDelStartPosReturnExt   OBJECT IDENTIFIER  ::=
{trackDataExtendedServiceParameters 1}
```

4.11.8 TRANSFER SYNTAX

The Procedure PDU Module annex shall contain a subsection specifying the transfer syntax to be used for the extension types defined in the Procedure PDU Module.

NOTES

1 If the transfer syntax uses encoding rules for ASN.1 that are defined in an external specification (e.g., an ISO standard), the specification of the transfer syntax can simply consist of a reference to that external document, e.g., 'The extension types specified in this module shall be encoded for transfer using the Basic Encoding Rules specified in [<reference number for the BER specification>]'.

2 Specification and use of transfer syntax encoding rules that are defined specifically for the derived service are outside the scope of this version of the Guidelines.

4.12 OBJECT IDENTIFIERS OF PROCEDURE PARAMETERS, EVENTS, AND DIRECTIVES

If as part of the specification of a CSTS for derived or service-original procedures new parameters, events and/or directives are specified, Object Identifiers need to be assigned so that those parameters, events and/or directives are accessible by the standard operations.

If required for the given CSTS, those assignments shall be captured in a dedicated ASN.1 module where the OID of this module shall be specified as follows:

```
CCSDS-<service identifier>-PROCEDURE-PARAMETERS-EVENTS-DIRECTIVES

{  iso(1) identified-organization(3) standards-producing-organization(112)
ccsds(4) css(4) csts(4) services(2) <service identifier>Service(<service
number>) <service identifier>ServiceModules(4) objectIdentifiers(1)
}
```

where:

a) <service identifier> is a character string used to name the CSTS type; and

b) <service number> is an integer that has been uniquely assigned to this CSTS type and registered with SANA.

The content of this module shall be equivalent to subsection F4.16 of the CSTS Specification Framework (reference [1]).

4.13 '<NAME OF SERVICE> PRODUCTION' ANNEX

4.13.1 If the CSTS requires any production functionality that is not defined in an existing standard, the CSTS specification shall include a normative annex entitled '<name of service> Production'.

NOTE – The specification of the production is documented in an annex instead of the main body of the CSTS specification because it is not formally part of the transfer service (although, of course, the transfer service depends on the production).

4.13.2 The '<name of service> Production' annex shall define the function(s) performed by the service provider and the Functional Resources that are required to produce or consume the data that is transferred by the CSTS.

4.13.3 The '<name of service> Production' annex shall identify and define any configuration parameters that are needed to configure the production process or that otherwise must be established in order to support the proper operation of the production process.

4.13.4 To the extent that it can be determined at the time that the service specification is written, the specification of each configuration parameter should include any requirements related to the degree to which the parameter value must be reconfigurable with respect to the service agreement or service package (e.g., the configuration parameter value applies to all instances in all service packages, or the parameter value must be configurable on a per-service package basis but remains static for the duration of the service package).

4.13.5 The ‘<name of service> Production’ annex shall identify the monitored parameters (if any) of the Functional Resources associated with the production process used if available to determine the `production-status` parameter value.

4.13.6 The ‘<name of service> Production’ annex shall identify the notifiable events (if any) of the Functional Resources associated with the production process that if available are used to determine service type specific events. The definition of the derived events shall include the `event-name` and `event-value` (if any) specification of each service type specific notifiable event.

4.13.7 The ‘<name of service> Production’ annex shall identify the name and Object Identifier of the Functional Resource Type that models the <name of service> provider. To that end, each service type specific OID subtree contains a reference to the OID of the associated service-type specific Functional Resource that is registered as an element of the subtree under the ‘crossSupportFunctionalities’ node.

4.14 IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA

The CSTS specification shall contain a normative Implementation Conformance Statement Proforma annex.

This annex shall provide the Implementation Conformance Statement (ICS) Requirements List (RL) for an implementation of the provider of the service.

4.15 SECURITY, SANA, AND PATENT CONSIDERATIONS

The CSTS specification shall contain an informative annex addressing Security, SANA, and patent considerations in accordance with reference [3]. Depending on the specific CSTS it may be sufficient to adopt the corresponding annex H of reference [1]. If not, the specific requirements of the given CSTS type shall be addressed in this annex.

4.16 ACRONYMS

The CSTS specification shall contain a list with all acronyms and the corresponding full text used in the Recommended Standard.

4.17 INFORMATIVE REFERENCES

The CSTS specification shall contain an informative annex listing informative references providing information useful in the context of the CSTS, but not containing binding specifications.

4.18 IMPLEMENTATION OF A CROSS REFERENCE MATRIX TO THE SERVICE SPECIFICATION FRAMEWORK

The CSTS shall provide in an informative annex a table listing the specific (sub)sections and paragraphs of the CSTS Specification Framework (reference [1]) that are referenced by the new CSTS, and identifies the (sub)sections and paragraphs of this new CSTS that make specific reference to each of those CSTS Specification Framework (sub)sections/paragraphs.

NOTE – Annex F of this Recommended Practice is an example of such annex.

ANNEX A

SECURITY, SANA AND PATENT CONSIDERATIONS

(INFORMATIVE)

The security, SANA, and patent considerations that are applicable to CSTS specifications developed based on these Guidelines are specified in annex H of reference [1].

ANNEX B

MAPPING OF EXTENSIONS TO CSTS SPECIFICATION FRAMEWORK ASN.1 DATA STRUCTURES

(INFORMATIVE)

This annex summarizes the extension points found in the CSTS Specification Framework operations for: (a) adding parameters to operation invocations, returns, and acknowledgments; (b) adding new `diagnostic` values types to the `diagnostic` parameters; (c) adding new notifiable events and associated event values to the `event-name` and `event-value` parameters; (d) adding new directives and the associated directive qualifiers to the `directive-identifier` and the `directive-qualifier` parameters; and (e) adding parameter type values to the `qualified-parameters` parameter.

B1 EXTENDED TYPE

As specified in the CSTS Specification Framework (reference [1]), each extension parameter is defined to be of type `Extended`, which is defined as follows:

```
Extended ::= CHOICE
{  external [0] Embedded
,  notUsed  [1] NULL
}

Embedded ::=   EMBEDDED PDV
```

If the extension is used, the choice [0] is to be used. The `EMBEDDED PDV` is an ASN.1 type that gives the capability to the specifier

a) to identify the syntax (in the form of an Object Identifier); and

b) to carry the value associated to the defined syntax.

If the extension is not used, the choice [1] is to be selected.

The `EMBEDDED PDV` that is used in the CSTS Specification Framework (reference [1]) is defined as follows:

```
EMBEDDED PDV ::= [UNIVERSAL 11] SEQUENCE
{  identification CHOICE
   {  syntaxes SEQUENCE
      {  abstract    OBJECT IDENTIFIER
      ,  transfer    OBJECT IDENTIFIER
      }
   ,  syntax   OBJECT IDENTIFIER
   ,  presentation-context-id INTEGER
   ,  context-negotiation  SEQUENCE
      {  presentation-context-id INTEGER
      ,  transfer-syntax OBJECT IDENTIFIER
      }
   ,  transfer-syntax      OBJECT IDENTIFIER
   ,  fixed          NULL
   }
,  data-value  OCTET STRING
}
```

NOTE – The use of the `EMBEDDED PDV` can be found in F2.2 of the CSTS Specification Framework (reference [1]).

B2 ADDING PARAMETERS TO OPERATIONS

B2.1 INVOCATIONS

B2.1.1 If a CSTS derives an operation from a common operation of the CSTS Specification Framework, parameters may be added to the invocation for that operation via the `<operation>InvocExt` parameter of the invocation for that operation, as defined in annex F of the CSTS Specification Framework (reference [1]).

NOTE – All invocations for the common operation are formatted in the CSTS Specification Framework following the template:

```
<Operation>Invocation      ::=   SEQUENCE
{  standardInvocationHeader        StandardInvocationHeader
[, zero or more parameters]
,  <operation>InvocExt             Extended
}
```

B2.1.2 If a CSTS derives an operation from a CSTS Specification Framework procedure-specific operation that already has added invocation parameters (with respect to the common operation), additional parameters may be added to the invocation via the `<derivedOperation>InvocExt` parameter that is defined for that operation invocation in annex F of the CSTS Specification Framework (reference [1]). Table B-1 identifies the CSTS Specification Framework procedure-specific operations for which the invocations have added parameters. If a CSTS adds parameters to one of these operation invocations, it must do so by extending the corresponding `<derivedOperation>InvocExt` parameter of the extension type.

Table B-1: Extension Parameters for Adding Parameters to CSTS Specification Framework Procedure Operation Invocations, Returns, and Acknowledgements

Procedure	Operations	Extended	Extension Type
Association Control	BIND	N	
	UNBIND	N	
	PEER-ABORT	N	
Unbuffered Data Delivery	START	N	
	STOP	N	
	TRANSFER-DATA	N	
Buffered Data Delivery	START	Y	BuffDataDelStartInvocExt
	STOP	N	
	TRANSFER-DATA	N	
	NOTIFY	Y	(Refer to B4)
Data Processing	START	N	
	STOP	N	
	PROCESS-DATA	Y	DataProcProcDataInvocExt
	NOTIFY	Y	(Refer to B4)
Buffered Data Processing	START	N	
	STOP	N	
	PROCESS-DATA	N	
	NOTIFY	N	(Refer to B4)
Sequence-Controlled Data Processing	START	Y	SequContrDataProcStartInvocExt
	STOP	N	
	PROCESS-DATA	Y	SequContrDataProcProcDataInvocExt SequContrDataProcProcDataPosReturnExt SequContrDataProcProcDataNegReturnExt
	NOTIFY	Y	(Refer to B4)
	EXECUTE-DIRECTIVE	N	

Procedure	Operations	Extended	Extension Type
Information Query	GET	N	
Cyclic Report	START	Y	CyclicReportStartInvocExt
	STOP	N	
	TRANSFER-DATA	N	Refined: CyclicReportTransferDataInvocDataRef
Notification	START	Y	NotificationStartInvocExt
	STOP	N	
Throw-Event	EXECUTE-DIRECTIVE	Y	TeExecDirNegReturnDiagnosticExt

B2.2 RETURNS AND ACKNOWLEDGEMENTS

NOTE – All returns and acknowledgements for the common operations are of type StandardReturnHeader, which is defined in F4.3 of the CSTS Specification Framework (reference [1]) as having the structure:

```
StandardReturnHeader      ::=   SEQUENCE
{  performerCredentials       Credentials
,  invokeId                   InvokeId
,  result                     CHOICE
   {  positive [0]   Extended -- To carry the positive results
   ,  negative [1]   SEQUENCE
      {  diagnostic     Diagnostic
      ,  negExtension   Extended
                                              -- The default
value of the
                        -- negExtension parameter is
                        -- 'notUsed'.
                        -- Unless a PDU that uses the
                        -- StandardReturnHeader explicitly
                        -- defines an extension type to be used
                        -- as the value of negExtension for
                        -- that PDU, the value shall be
                        -- 'notUsed'.
      }
   }
}
```

B2.2.1 If a CSTS derives an operation from a common operation of the CSTS Specification Framework, a parameter may be added to the positive result return or positive result acknowledgement via the `positive` choice of the `result` parameter of the return/acknowledgement for that operation, as defined in annex F of the CSTS Specification Framework (reference [1]).

NOTE – The Sequence-Controlled Data Processing procedure extends the positive return of the PROCESS-DATA operation by adding a parameter. The type specifying this extension is `SequContrDataProcProcDataPosReturnExt` (see table B-1).

B2.2.2 If the CSTS derives an operation from a common operation of the CSTS Specification Framework, a parameter may be added to the negative result return or negative result acknowledgement via the `negExtension` parameter of the `negative` choice of the `result` parameter of the return / acknowledgement for that operation, as defined in annex F of the CSTS Specification Framework (reference [1]).

NOTE – The Sequence-Controlled Data Processing procedure extends the negative return of the PROCESS-DATA operation by adding a parameter. The type specifying this extension is `SequContrDataProcProcDataNegReturnExt` (see table B-1).

B3 EXTENSION VALUES FOR THE `diagnostic` PARAMETER

B3.1 If the CSTS derives an operation from a common operation of the CSTS Specification Framework that has a return or acknowledgement, additional `diagnostic` values may be added to the negative result for the return or acknowledgement via the `diagnosticExtension` choice of the `diagnostic` parameter of the `negative` choice of the `result` parameter of the return or acknowledgement for that operation, as defined in annex F of the CSTS Specification Framework (reference [1]).

NOTE – All `diagnostic` parameters are of the type `Diagnostic`, which is defined in F4.3 of the CSTS Specification Framework (reference [1]) as:

```
Diagnostic           ::=   CHOICE
{  invalidParameterValue   [1]   SEQUENCE
   {  text        AdditionalText
   ,  appellation Appellation -- of the invalid parameter
   }
,  conflictingValues       [2]   SEQUENCE
   {  text           AdditionalText
   ,  appellations   SEQUENCE OF Appellation
   }
,  otherReason             [3]   AdditionalText
,  unsupportedOption       [4]   AdditionalText
,  diagnosticExtension     [100] Embedded
}
```

B3.2 Some of the common operations as defined in the CSTS Specification Framework (reference [1]) specify operation type specific extensions of the `diagnostic` parameter as shown in table B-2. Each of these extensions are specified such that further extensions are possible if needed in a procedure-specific context.

Table B-2: Extension Parameters for Adding `diagnostic` Values to CSTS Specification Framework Common Operation Negative Returns and Negative Acknowledgements

Operation	Extended	Extension Type
BIND	Y	AssocBindDiagnosticExt (see NOTE)
UNBIND	N	
START	Y	StartDiagnosticExt
STOP	N	
PROCESS-DATA	N	
GET	Y	GetDiagnosticExt
EXECUTE-DIRECTIVE	Y	ExecDirNegAckDiagnosticExt ExecDirNegReturnDiagnosticExt
NOTE – The BIND return is a special case in the sense that the operation may only be used for the Association Control procedure. For that reason the BIND operation types are specified as part of the Association Control procedure in F4.5 of the CSTS Specification Framework (reference [1]) and not as part of the 'common' operations. Therefore also the extension of the Diagnostic type is named as if the extension were procedure-type specific, although strictly speaking the extension is specific for the negative return of the BIND operation.		

B3.3 If a CSTS derives an operation from a CSTS Specification Framework procedure-specific operation that already has added `diagnostic` parameter values for the return (with respect to the common operation), additional `diagnostic` values may be added to the return via the `diagnosticExtension` parameter that is defined for that operation in annex F of the CSTS Specification Framework (reference [1]). Table B-3 identifies the CSTS Specification Framework procedure-specific operations for which the returns have added `diagnostic` values. If a CSTS adds `diagnostic` values to one of these operation invocations, it must do so by extending the corresponding `extensionDiagnostic` of the `Embedded` type.

Table B-3: Extension Parameters for Adding `diagnostic` Values to CSTS Specification Framework Procedure Operation Negative Returns

Procedure	Operations	Extended	Extension Type
Association Control	BIND	(See NOTE below table 4-2)	AssocBindDiagnosticExt
Buffered Data Delivery	START	Y	BuffDataDelStartDiagnosticExt
Sequence-Controlled Data Processing	PROCESS-DATA	Y	SequContrDataProcProcDataDiagnosticExt
Cyclic Report	START	Y	CyclicReportStartDiagnosticExt
Notification	START	Y	NotificationStartDiagnosticExt
Throw-Event	EXECUTE-DIRECTIVE	Y	TeExecDirNegReturnDiagnosticExt

B4 `NOTIFYINVOCATION` – EVENTS HANDLING

B4.1 If the CSTS creates a procedure that uses the common NOTIFY operation or adds a common NOTIFY operation to a procedure, additional notifiable events may be added via the `eventName` parameter of the `NotifyInvocation` type, as defined in F4.4 of the CSTS Specification Framework (reference [1]). A new added notifiable event has to follow the definition of the `eventName` and `eventValue` parameters.

NOTES

1 The `NotifyInvocation` type is defined in F4.4 of the CSTS Specification Framework (reference [1]) as:

```
NotifyInvocation        ::=   SEQUENCE
{  standardInvocationHeader   StandardInvocationHeader
,  eventTime                  Time
,  eventName                  Name
,  eventValue                 EventValue
,  notifyInvocationExtension  Extended
}
```

2 The following procedure independent published Event Identifiers (`eventName`) are defined in F4.17 of the CSTS Specification Framework (reference [1]):

```
- svcProductionStatusChange
- svcProductionConfigurationChange
```

B4.2 A CSTS making use of the NOTIFY operation may add additional events by defining `eventName` and `eventValue` parameter arguments. Table B-4 identifies the CSTS Specification Framework procedure-specific NOTIFY operations that have added notifiable events to the `NotifyInvocation`.

NOTE – Table B-4 lists the notifiable events that have been added for specific procedures specified in the CSTS Specification Framework (reference [1]).

Table B-4: Adding Notifiable Events to CSTS Specification Framework Procedure NOTIFY Invocations

Procedure	Added Notifiable Event
Buffered Data Delivery	pBDDdataDiscardedExcessBacklog pBDDrecordingBufferOverflow pBDDendOfData pBDDconfigurationChange
Data Processing	pDPdataProcessingCompleted pDPconfigurationChange
Sequence-Controlled Data Processing	pSCDPexpired pSCDPlocked

B4.3 If a CSTS creates a procedure that uses the NOTIFY operation, adds a NOTIFY operation to a procedure, or uses a CSTS Specification Framework procedure that contains a NOTIFY operation, additional content may be added to any or all of the published events by defining an extension type using the `notificationInvocationExtension` parameter of the `NotifyInvocation` type as defined in F4.4 of the CSTS Specification Framework (reference [1]).

NOTE – Table B-5 lists the NOTIFY invocation extension that has been specified for the Data Processing procedure defined in F4.8 of the CSTS Specification Framework (reference [1]). The Buffered Data Processing procedure and the Sequence-Controlled Data Processing Procedure inherit this extension.

Table B-5: Procedure-Specific NOTIFY Invocation Extensions

Procedure	NOTIFY Invocation Extension
Data Processing	DataProcNotifyInvocExt

B5 EXTENSION VALUES FOR QUALIFIED-PARAMETERS PARAMETER

B5.1 If the CSTS uses the common GET operation from the CSTS Specification Framework, additional type values for the qualified-parameters parameter may be added by defining an extension type for the typeAndValueExtension choice of the TypeAndValue type of the TypeAndValueComplexQualified choice, as defined in F4.3 of the CSTS Specification Framework (reference [1]).

NOTE – The CSTS Specification Framework (reference [1]) defines the qualified-parameters parameter of the GET operation to be a sequence of QualifiedParameter, where QualifedParameter is defined as:

```
QualifiedParameter  ::=  SEQUENCE
{ parameterName              Name
, qualifiedValues            SEQUENCE OF QualifiedValues
}

QualifiedValues  ::=  CHOICE
{ valid       [0]   TypeAndValueComplexQualified -- Valid value
, unavailable [1]   NULL  -- Unknown or unavailable value
, undefined   [2]   NULL  -- Undefined in the context
, error       [3]   NULL  -- Processing resulted in an error
}
```

and TypeAndValueComplexQualified type is defined in the CSTS Specification Framework as:

```
TypeAndValueComplexQualified ::=  CHOICE
{ typeAndValue      [0]   TypeAndValue
, complexSequence   [1]   SEQUENCE OF TypeAndValue
, complexSet        [2]   SET OF TypeAndValue
}
```

and TypeAndValue type is defined in the CSTS Specification Framework as:

```
TypeAndValue        ::=   CHOICE
{ integer                 [0]   SEQUENCE OF INTEGER
, integerPositive         [1]   SEQUENCE OF IntPos
, intUnsigned             [2]   SEQUENCE OF IntUnsigned
, duration                [3]   SEQUENCE OF Duration
, characterString         [4]   SEQUENCE OF VisibleString
, boolean                 [5]   SEQUENCE OF BOOLEAN
, octetString             [6]   SEQUENCE OF OCTET STRING
, float                   [7]   SEQUENCE OF REAL
, time                    [8]   SEQUENCE OF Time
, enumerated              [9]   SEQUENCE OF INTEGER
, publishedIdentifier     [10]  SEQUENCE OF PublishedIdentifier
, typeAndValueExtension   [100] Embedded
}
```

B5.2 If the CSTS derives a TRANSFER-DATA operation from the CSTS Specification Framework Cyclic Report procedure, additional type values for the `qualifiedParameters` parameter of the `CyclicReportTransferDataInvocDataRef` type may be added by defining an extension type for the `TypeAndValueExtension` choice of the `TypeAndValue` type (see note under B5.1), as defined in F4.3 of the CSTS Specification Framework (reference [1]).

B6 ABSTRACT PARAMETERS

B6.1 GENERAL

B6.1.1 Abstract parameters in the CSTS Specification Framework ASN.1 are structured as choices between an octet-string value and an extension parameter of type `EMBEDDED PDV` (see `AbstractChoice` as defined in F4.3 of the CSTS Specification Framework (reference [1]).

B6.1.2 The CSTS Specification Framework (reference [1]) has two instances of abstract parameters: the `data` parameter of the TRANSFER-DATA invocation and the `data` parameter of the PROCESS-DATA invocation.

NOTE – The CSTS Specification Framework does not use the term ‘abstract parameter’. This term is introduced in this document as a name for the generalization of the `data` parameter of the PROCESS-DATA and TRANSFER-DATA operations.

B6.2 EXTENSION OF THE TRANSFER-DATA INVOCATION `data` PARAMETER

B6.2.1 If the CSTS uses the common TRANSFER-DATA operation from the CSTS Specification Framework (reference [1]), the `data` parameter of the invocation may be resolved by either (a) refining the value of the `opaqueString` choice of the `data` parameter or (b) defining an extension type for the `extendedData` choice of the `data` parameter of the common `TransferDataInvocation` type, as defined in F4.4 of the CSTS Specification Framework (reference [1]).

NOTE – As specified in F4.4 of the CSTS Specification Framework (reference [1]), the common TRANSFER-DATA invocation is defined as:

```
TransferDataInvocation ::=   SEQUENCE
{  standardInvocationHeader         StandardInvocationHeader
,  generationTime                   Time
,  sequenceCounter                  IntUnsigned
,  data                             AbstractChoice
,  transferDataInvocationExtension  Extended
}
```

where:

```
AbstractChoice ::=   CHOICE
{  opaqueString   [0]   OCTET STRING
,  extendedData   [1]   Embedded
}
```

B6.2.2 If the CSTS uses of the CSTS Specification Framework Buffered Data Delivery or Unbuffered Data Delivery procedure, the `data` parameter of the TRANSFER-DATA invocation may be resolved by either (a) refining the value of the `opaqueString` choice of the `AbstractChoice` of the `data` parameter or (b) defining an extension type for the `extendedData` choice of the `AbstractChoice` of the `data` parameter of the common `TransferDataInvocation`, as defined in F4.4 of the CSTS Specification Framework (reference [1]).

B6.2.3 The `data` parameter of the TRANSFER-DATA invocation of the CSTS Specification Framework Cyclic Report procedure is already resolved by refinement as the `qualifiedParameter` choice of the `CyclicReportTransferDataInvocDataRef` type. If a CSTS uses the Cyclic Report procedure, additional content may be added to the `data` parameter via the `cyclicReportTransferDataInvocDataRefExtension` choice of the `CyclicReportTransferDataInvocDataRef` type, as defined in F4.12 of the CSTS Specification Framework (reference [1]).

NOTE – As specified in F4.12 of the CSTS Specification Framework (reference [1]), the `CyclicReportTransferDataInvocDataRef` type is defined as:

```
CyclicReportTransferDataInvocDataRef  ::=   SEQUENCE
{  qualifiedParameters        SEQUENCE OF QualifiedParameter
,  cyclicReportTransferDataInvocDataRefExtension         Extended
}
```

B6.3 EXTENSION OF THE PROCESS-DATA INVOCATION `data` PARAMETER

B6.3.1 If the CSTS uses the common PROCESS-DATA operation from the CSTS Specification Framework, the `data` parameter of the invocation may be made concrete by either (a) refining the value of the `opaqueString` choice of the `AbstractChoice` of the `data` parameter or (b) defining an extension type for the `extendedData` choice of the `AbstractChoice` of the `data` parameter of the common `ProcessDataInvocation`, as defined in F4.4 of the CSTS Specification Framework (reference [1]).

NOTE – As specified in F4.4 of the CSTS Specification Framework (reference [1]), the common PROCESS-DATA invocation is defined as:

```
ProcessDataInvocation  ::=  SEQUENCE
{  standardInvocationHeader         StandardInvocationHeader
,  dataUnitId                       DataUnitId
,  data                             AbstractChoice
,  processDataInvocationExtension   Extended
}
```

where:

```
AbstractChoice          ::=  CHOICE
{  opaqueString      [0]         OCTET STRING
,  extendedData      [1]         Embedded
}
```

B6.3.2 If the CSTS uses the CSTS Specification Framework Data Processing procedure, the `data` parameter of the PROCESS-DATA invocation may be made concrete by either (a) refining the value of the `opaqueString` choice of the `AbstractChoice` of the `data` parameter or (b) defining an extension type for the `extendedData` choice of the `AbstractChoice` of the `data` parameter of the common `ProcessDataInvocation`, as defined in F4.4 of the CSTS Specification Framework (reference [1]).

NOTE – The Data Processing procedure does not extend the `data` parameter of the PROCESS-DATA operation.

B6.4 EXTENSION OF THE EXECUTE-DIRECTIVE INVOCATION `DIRECTIVE-IDENTIFIER` AND `DIRECTIVE-QUALIFIER` PARAMETERS

B6.4.1 If the CSTS uses the common EXECUTE-DIRECTIVE operation as defined in F4.4 of the CSTS Specification Framework (reference [1]), the `directive-identifier` parameter of the invocation must be Published Identifier registered with SANA.

NOTE – As specified in F4.4 of the CSTS Specification Framework (reference [1]), the common EXECUTE-DIRECTIVE invocation is defined as:

```
ExecuteDirectiveInvocation    ::=   SEQUENCE
{  standardInvocationHeader           StandardInvocationHeader
,  directiveIdentifier                PublishedIdentifier
,  directiveQualifier        CHOICE
   {  localProcDirQualifier        [1]DirectiveQualifierValues
   ,  serviceProcDirQualifier      [2]SEQUENCE
      {  procedureInstanceId          ProcedureInstanceId
      ,  serviceProcDirQualifierValues DirectiveQualifierValues
      }
   ,  functResourceDirQualifier    [3]SEQUENCE
      {  functResourceInstanceNumber
                             FunctionalResourceInstanceNumber
      ,  functionalResourceQualifiers  DirectiveQualifierValues
      }
   ,  directiveQualifierExtension  [4]Embedded
   }
,  executeDirectiveInvocationExtension       Extended
}
```

B6.4.2 If the CSTS uses the common EXECUTE-DIRECTIVE operation from the CSTS Specification Framework, the `directive-qualifier` parameter of the invocation must be either (a) specified to be a set of Parameter Labels with their associated values if the directive acts on the procedure that contains the EXECUTE-DIRECTIVE operation (the applicable choice is `localProcDirQualifier`), or (b) defined to be a sequence of a procedure instance identifier and a set of Parameter Labels with their associated values if the directive acts on a specific procedure being part of the CSTS (the applicable choice is `serviceProcDirQualifier`), or (c) defined to be a sequence of a Functional Resource Instance identifier and a set of Parameter Labels with their associated values if the directive acts on a specific Functional Resource being part of the CSTS (the applicable choice is `functResourceDirQualifier`), or (d) an extension type in case none of the options (a), (b) and (c) applies (the applicable choice is `directiveQualifierExtension`).

ANNEX C

EXAMPLE SPECIFICATION OF A CSTS WITH DERIVED OPERATION PARAMETERS

(INFORMATIVE)

C1 INTRODUCTION

This annex presents an example of the Composition and Derived Procedures sections of the service specification for a MyNewService CSTS that derives a Non-Sense procedure from the CSTS Specification Framework Buffered Data Delivery procedure as follows:

a) the START operation adds parameters to allow the service user to select the gender (male, female, hermaphrodite or neutral) and age (0- 99) of the source of the data; and

b) the data parameter of the TRANSFER-DATA invocation is refined to carry the current color of the data source (red, blue, white, or black).

The new service is completely contrived for the purpose of illustrating derivation. The extension parameters and their values have no relationship to real-world spaceflight operations or the data flows associated with them.

The following CSTS example represents the text for sections 3 (Composition) and 4 (Derived Procedures) of the MyNewService specification, and follows the format and content as stipulated by this Recommended Practice. Throughout this text, explanatory comments are contained in *[square-bracketed italicized text]*. The content of these explanatory comments would not appear in a real CSTS specification.

C2 MYNEWSERVICE EXAMPLE SPECIFICATION (COMPOSITION AND DERIVED PROCEDURES SECTIONS)

3 COMPOSITION OF THE MyNewService CROSS SUPPORT TRANSFER SERVICE

3.1 OVERVIEW

The MyNewService service is a concrete service. That is, it can be implemented as defined herein without need for further extension or refinement.

3.2 COMPONENT PROCEDURES OF THE MyNewService SERVICE

3.2.1 The MyNewService service shall be composed of the procedures identified in table 3-1. The elements of that table are defined in the 'COMPONENT PROCEDURES SUBSECTION' subsection of the CSTS Guidelines (reference [x]).

Table 3-1: MyNewService Component Procedures

Procedure	Association Control	Non-Sense [P]
Version	-	1
No. of Instances	1..1	1..1
Specification Approach	adopted	derived
Source	Subsection 4.3 of reference [y]: Association Control	Subsection 4.5 of reference [y]: Buffered Data Delivery

3.2.2 The MyNewService service shall use the Association Control procedure as defined in the CSTS Specification Framework (reference [y]) without derivation.

3.2.3 The MyNewService service shall use the Non-Sense procedure, which is derived from its parent procedure type.

3.3 MyNewService STATE MACHINE

The MyNewService service state machine shall conform to the state machine for a CSTS with stateful prime procedure instance, as defined in the CSTS Specification Framework (reference [y]).

4 NON-SENSE

4.1 DISCUSSION

[The 'Description' subsection contains 'Purpose' and 'Concept' subsections.]

4.2 PROCEDURE TYPE IDENTIFIER

The Object Identifier for the derived Non-Sense procedure of the MyNewService service shall be defined as:

```
nonSenseProcedure  OBJECT IDENTIFIER  ::= {myNewSvcServiceProcedures  1}
```

4.3 EXTENSION

The Non-Sense procedure extends the Buffered Data Delivery procedure by addition of parameters to the START operation and refinement of the `data` parameter of the TRANSFER-DATA operation.

4.4 BEHAVIOR

The behavior of the Non-Sense procedure is identical to that of the Buffered Data Delivery procedure, with the following exceptions:

4.4.1 <Description of the behavior of the provider with respect to the setting of the `gender` and `age` parameters of the START invocation.>

4.4.2 <Description of the process by which the color value that is delivered in the `data` parameter of the TRANSFER-DATA invocation is determined.>

4.5 REQUIRED OPERATIONS

The operations of the Non-Sense procedure are those of the Buffered Data Delivery procedure, with the following extensions and refinements:

Table 3-2: Non-Sense Procedure Required Operations

Operations	Extended	Refined	Procedure Blocking/Non-Blocking
START	Y	N	Blocking
STOP	N	N	Blocking
TRANSFER-DATA	N	Y	Non-Blocking
NOTIFY	N	N	Non-Blocking

4.5.1 START

4.5.1.1 Invocation, Return, and Parameters

The common parameters of the START operation of the Buffered Data Delivery Procedure are defined in the CSTS Specification Framework (reference [y]). Table 3-3 identifies the extension parameters of the Non-Sense START operation.

Table 3-3: Non-Sense START Operation Extension Parameters

Extension Parameters	Invocation	Return
gender	M	
age	M	

4.5.1.1.1 `gender`

The `gender` parameter shall contain the gender of the data source that is to be used as the source of data for the Cross Support Transfer Service instance. The valid values are: 'male', female', 'hermaphrodite', and 'neutral'.

4.5.1.1.2 `age`

The `age` parameter shall contain the age (in years) of the data source that is to be used as the source of data for the Cross Support Transfer Service instance. The valid values are in the range 0-99 inclusive.

4.5.1.2 Extension Parameters Syntax

The type `NonSenseStartInvocExt`, as defined in annex X, shall define the syntax and the transfer syntax of the extension parameter of the START invocation.

4.5.2 TRANSFER-DATA

4.5.2.1 Invocation, Return, and Parameters

The common parameters of the TRANSFER-DATA operation of the Buffered Data Delivery Procedure are defined in the CSTS Specification Framework (reference [1]).

4.5.2.1.1 `data` Parameter Refinement

[The common TRANSFER-DATA invocation is specified in the CSTS Specification Framework as:

```
TransferDataInvocation       ::=   SEQUENCE
{  standardInvocationHeader           StandardInvocationHeader
,  generationTime                     Time
,  sequenceCounter                    IntUnsigned
,  data                               AbstractChoice
,  transferDataInvocationExtension    Extended
}
```

where:

```
AbstractChoice          ::=   CHOICE
{  opaqueString      [0]         OCTET STRING
,  extendedData      [1]         Embedded
}
```

The refinement consists of specifying the syntax and semantic of the `opaqueString` component of the `data` parameter.]

4.5.2.1.1.1 The `data` parameter shall be refined to represent one of four colors: red, blue, white, or black.

4.5.2.1.1.2 The `data` parameter shall be encoded using the `opaqueString` choice.

4.5.2.1.1.3 The `data` parameter shall be encoded on one octet having one of the following hexadecimal values:

– 01x: red;

– 02x: blue;

– 10x: white;

– 20x: black.

4.6 CONFIGURATION PARAMETERS

The Non-Sense procedure adopts the configuration parameters of the Buffered Data Delivery procedure without addition or modification

4.7 PROCEDURE STATE TABLES

The Non-Sense procedure adopts the state tables of the Buffered Data Delivery procedure without modification.

ANNEX X

X.1 Syntax

The **data-value** parameter of the Embedded PDV for the START invocation extension parameter has the syntax:

```
<xs:complexType name="NonSenseStartInvocExt">
      <xs:sequence>
            <xs:element name="gender" type="Gender" />
            <xs:element name="age" type="NumericString"/>
            <xs:element name="extensionParameter" type="xs:hexBinary"/>
      </xs:sequence>
</xs:complexType>

<xs:simpleType name="Gender">
      <xs:restriction base="xs:string">
            <xs:enumeration value="male"/>
            <xs:enumeration value="female"/>
            <xs:enumeration value="hermaphrodite"/>
            <xs:enumeration value="neutral"/>
      </xs:restriction>
</xs:simpleType>

<xs:simpleType name="NumericString ">
      <xs:restriction base="xs:token">
            <xs:pattern value="\d{2}"/>
      </xs:restriction>
/xs:simpleType>
```

NOTE – The XML document compliant to this schema needs to be UTF-8 encoded.

X.2 ALLOWED VALUES

X.2.1 OVERVIEW

The transfer syntax of the **data-value** parameter of the `EMBEDDED PDV` for the START invocation extension parameter is specified in X.2.2.

X.2.2 TRANSFER SYNTAX OF THE DATA-VALUE PARAMETER OF THE EMBEDDED PDV FOR THE START INVOCATION EXTENSION PARAMETER

X.2.2.1 The NonSenseStartInvocExt shall be encoded in an ASCII string.

X.2.2.2 The first characters of the ASCII string shall be ‘male’, ‘female’, ‘hermaphrodite’, or ‘neutral’, corresponding to the value of the `gender` parameter.

X.2.2.3 The character immediately following the last character of the gender subfield shall be a comma (‘,’).

X.2.2.4 The two characters immediately following the comma shall represent the age parameter.

X.2.2.5 For the Non-Sense procedure of the myNewService service, the ASCII string value shall terminate after the second numerical character of the age subfield of the string.

X.2.2.6 Procedures derived from the Non-Sense procedure may add extension parameters as follows:

a) for each extension parameter of the derived procedure, the valid ASCII values shall be specified;

b) the sequence of the derived procedure extension parameters in the **data-value** ASCII string shall be specified;

c) a comma shall appear immediately after the last character of the **age** subfield in the ASCII string;

d) the first extension parameter for the derived procedure shall appear immediately after the comma in the ASCII string;

e) for each additional derived procedure extension parameter, a comma shall appear immediately after the last character of the previous extension parameter value, followed by the value of the next extension parameter, in the specified order.

X.3 Transfer Syntax Identifier

The value of the **syntax** parameter of the Embedded PDV for the START invocation extension parameter is specified as:

```
NonSenseStartInvocExt OBJECT IDENTIFIER ::=

      {myNewSvcExtendedServiceParameters 1}
```

ANNEX D

ACRONYMS

(INFORMATIVE)

ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
CCSDS	Consultative Committee for Space Data Systems
CSTS	Cross Support Transfer Services
ICS	Implementation Conformance Statement
ISO	International Organization for Standardization
M	mandatory
N	no
OID	object identifier
PDU	protocol data unit
PDV	Presentation Data Value (ASN.1)
RL	requirements list
SANA	Space Assigned Numbers Authority
SLE	Space Link Extension
Y	yes

ANNEX E

INFORMATIVE REFERENCES

(INFORMATIVE)

[E1] *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.

[E2] *Cross Support Service Management—Service Management Utilization Request Formats*. CCSDS 902.9. Proposed.

[E3] *Cross Support Service Management—Simple Schedule Format Specification*. Issue 2. Draft Recommendation for Space Data System Standards (Red Book), CCSDS 902.1-R-2. Washington, D.C.: CCSDS, May 2017.

[E4] *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.

[E5] *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.

[E6] *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.

ANNEX F

CROSS REFERENCES TO SPECIFIC (SUB)SECTIONS OF THE CSTS SPECIFICATION FRAMEWORK

(INFORMATIVE)

Table F-1 lists the specific sections and paragraphs of the CSTS Specification Framework (reference [1]) that are referenced by this Recommended Practice, and identifies the sections and subsections of this Recommended Practice that make specific reference to each of those CSTS Specification Framework sections/subsections.

Table F-1: Cross Reference to CSTS Specification Framework Sections and Subsections

CSTS Specification Framework SFW Section/ Subsection (reference [1])	Referencing Sections/Subsections of this Recommended Practice
1.6.1	4.3.6 a)
1.6.3.3	1.6.2
2.5.3	3.1.4
3.4.2.2.4.3	4.7
4.3	3.4.2, 3.4.2 NOTE, 4.5.3.5 table 4-1, 4.5.3.15 table 4-2, 4.5.3.16 table 4-3
4.3.5 table 4-2	4.7.2.4 NOTE 2
4.5	4.6.8.3.7.2 NOTE, 4.11.7.4.1, 4.11.7.4.1 NOTE
4.6.4.2	3.11.3.4.4 NOTE
4.8	3.11.4.8 NOTE
4.10	4.5.3.5 table 4-1, 4.5.3.15 table 4-2, 4.5.3.16 table 4-3, 4.6.8.3.4.1 NOTE
Annex B	4.8
Annex C	3.11.3.5.1, 4.6.8.3.10.2
Annex D	3.11.3.4.2, 4.10.5.1, 4.10.5.2, 4.10.5.4, 4.10.5.6, 4.10.5.8, 4.10.5.10, 4.10.5.12
D4	3.10
D5	3.3, 3.11.3.5.3 d), 3.11.4.6 e)

CSTS Specification Framework SFW Section/ Subsection (reference [1])	**Referencing Sections/Subsections of this Recommended Practice**
Annex F	3.11.3.2.2, 3.11.3.2.4, 3.11.3.2.6, 3.11.3.3.2, 3.11.3.4.2, 3.11.3.5.2, B2.1, B2.1.2, B2.2.1, B2.2.2, B3.1, B3.3
F2.2	B1
F4	1.5.1
F4.1	4.10.4.1
F4.3	3.5.5 NOTE, 3.11.3.4.2 NOTE 2, B2.2 NOTE, B3.1 NOTE, B5.1, B5.2, B6.1.1
F4.4	B4.1, B4.1 NOTE 1, B4.3, B6.2.1, B6.2.1 NOTE, B6.3.2, B6.4.1, B6.4.1 NOTE, B6.3.2, B6.5.1, B6.5.1 NOTE
F4.5	B3.2 NOTE
F4.8	B4.3 NOTE
F4.12	B6.3.3, B6.3.3 NOTE
F4.15	4.11.5.3.2
F4.16	4.12
F4.17	B4.1 NOTE 2
Annex G	3.2, 4.5.4.3, 4.5.4.4
Annex H	1.5.1, 4.15, annex A
H2.2	3.10
H2.3	3.3