The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

CROSS SUPPORT TRANSFER SERVICES—MONITORED DATA SERVICE

RECOMMENDED STANDARD

CCSDS 922.1-B-1

BLUE BOOK

April 2017

The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

CROSS SUPPORT TRANSFER SERVICES—MONITORED DATA SERVICE

RECOMMENDED STANDARD

CCSDS 922.1-B-1

BLUE BOOK

April 2017

AUTHORITY

Issue:	Recommended Standard, Issue 1
Date:	April 2017
Location:	Washington, DC, USA

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

This document is published and maintained by:

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

STATEMENT OF INTENT

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

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

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

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

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

FOREWORD

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

http://www.ccsds.org/

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

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

Member Agencies

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

Observer Agencies

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

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 922.1-B-1	Cross Support Transfer Services—Monitored Data Service, Recommended Standard, Issue 1	April 2017	Original issue

CONTENTS

CONTENTS (continued)

CONTENTS (continued)

CONTENTS (continued)

Tables Page

1 INTRODUCTION

1.1 PURPOSE OF THIS RECOMMENDED STANDARD

This Recommended Standard defines the Monitored Data Cross Support Transfer Service (CSTS), in conformance with the Cross Support Transfer Services Specification Framework Recommended Standard (reference [1]). The Monitored Data CSTS (MD-CSTS) is a service that allows a spaceflight mission to receive cyclic reports on, and to query the current values of, the parameters that are pertinent to Cross Support Services being provided by a Cross Support Complex. The Monitored Data service also allows a spaceflight mission to receive notifications of the occurrence of events of interest associated with the services that are being provided by a Cross Support Complex.

NOTE – The term 'Cross Support Complex', as used throughout this document, corresponds to the Earth Space Link Terminal (ESLT) defined in the Space Communications Cross Support Architecture Description Document (reference [G9]).

1.2 SCOPE

This Recommended Standard defines the Monitored Data service in terms of:

a) the CSTS procedures that constitute the service;

b) the extensions and refinements of the behavior of those CSTS procedures necessary to provide the transfer service;

c) the extensions and refinements of standard CSTS operations associated with each of the procedures;

d) the relationships among the procedures that constitute the service.

It does not specify:

a) individual implementations or products;

b) the implementation of entities or interfaces within real systems;

c) the methods or technologies required to measure the values of monitored parameters and to detect the occurrence of events of interest;

d) the methods or technologies required for communication;

e) the management activities necessary to schedule, configure, and control the MD-CSTS;

f) the specific parameters that are to be reported and events that are to be notified by the MD-CSTS.

1.3 APPLICABILITY

1.3.1 GENERAL

The applicability and limits of applicability of Cross Support Transfer Services in general, as described in reference [1], pertain to the Monitored Data service, with the addition of the conditions described in 1.3.2, below.

1.3.2 APPLICABILITY OF THIS RECOMMENDED STANDARD

This Recommended Standard is applicable to the implementation of real systems that monitor provision and production of space communication Cross Support Services for the purposes of generating cyclic status reports, generating notifications of changes in status in real time, and responding to queries of current values of operational parameters.

1.4 RATIONALE

The goal of this Recommended Standard is to create a standard for interoperability for the exchange of cross support service-related status information between the cross support elements of various space Agencies and the users of the Cross Support Services that they provide.

1.5 DOCUMENT ORGANIZATION

Section 2 describes the Monitored Data Cross Support Transfer Service in terms of:

- the role of Service Management with respect to the MD-CSTS;
- the allocation of production and provision of the MD-CSTS to Functional Resources;
- the cross support view of the MD-CSTS;
- the functional description of the production and provision of the service; and
- an operational scenario that illustrates some of the more significant aspects of the service.

Section 3 specifies the composition of the MD-CSTS. The service type identifier is declared, the procedures that constitute the service are identified, and the CSTS state machine that applies to the MD-CSTS is specified. Because the MD-CSTS is composed of procedures that are directly adopted from the CSTS Framework without extension, no further specification of the MD-CSTS is required.

Section 4 specifies the On-Change-Option Cyclic Report procedure, which is an extension and refinement of the Framework Cyclic Report Procedure.

Section 5 specifies the refinements of the Information Query procedure.

Section 6 specifies the refinements of the Notification procedure.

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

Section 8 specifies the Monitored Data Service-specific versions of the service-generic parameter and events that are defined in *Cross Support Transfer Service—Specification Framework* (reference [1]).

Section 9 specifies the refinements of the definitions of Framework parameters, events, and directives for the purposes of making them applicable to the MD-CSTS.

Annex A is the Implementation Conformance Statement Proforma for the MD-CSTS.

Annex B provides the formal specification of the ASN.1 Object Identifiers module for the Monitored Data transfer service.

Annex C provides the formal specification of the ASN.1 Protocol Data Unit module for the On-Change-Option Cyclic Report procedure of the Monitored Data transfer service.

Annex D defines the monitored data production process. In particular, it specifies how monitored data values are to be labeled so that, when transferred by MD-CSTS instances, the sources of the measurements are unambiguous.

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

Annex F describes an example set of monitored parameters, notifiable events, and their associated Functional Resource Types. These examples are modeled on the contents of the SANA Functional Resource Registry (reference [3]).

NOTE – The SANA Functional Resource Registry is the normative repository of all such definitions. The example entries in annex F are included to illustrate the concepts of the use of Functional Resources by the MD service. Any real implementation of the MD service must use the Functional Resource specifications found in the SANA Functional Resource Registry.

Annex G provides the list of informative references.

Annex H lists the acronyms used in this document.

Annex I identifies the specific reference-[1] sections that are cross referenced by the MD-CSTS Recommended Standard, and the sections of the MD-CSTS Recommended Standard that reference each of those sections or subsections in reference [1]. This annex is included for maintainability of this Recommended Standard through future changes in reference [1].

1.6 CROSS SUPPORT TRANSFER SERVICES DOCUMENTATION

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

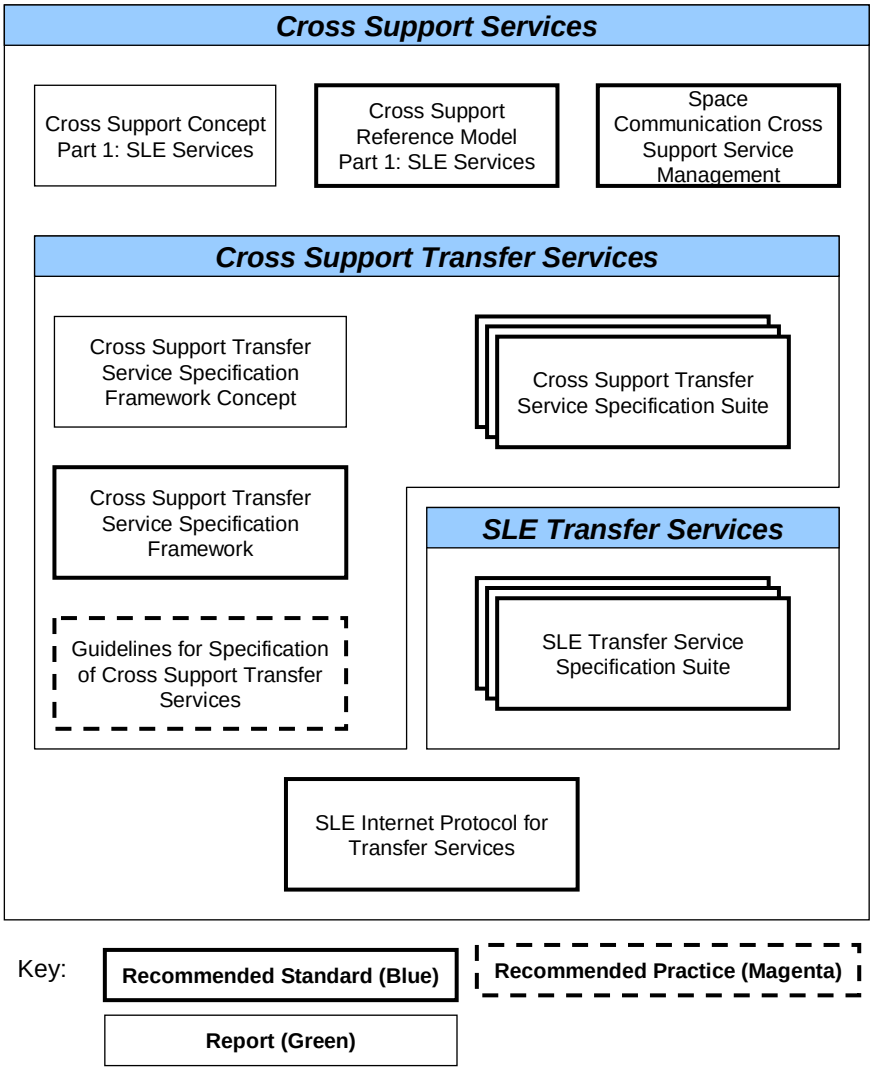

Figure 1-1: Cross Support Services Documentation

The CSTS Documentation is:

a) *Cross Support Concept—Part 1: Space Link Extension Services* (reference [G1]): a Report introducing the concepts of cross support and the Space Link Extension (SLE) services. Many of the concepts for the SLE transfer services have been adopted for the CSTSes (see h) below);

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

terminology for the specification of SLE services. Much of the framework and terminology of this reference model has been adopted or adapted for CSTSes (see 1.7.2 and 2.2).

c) *Space Communication Cross Support Service Management* suite (informative references [G6], [G7], and [G8]): Recommended Standards that specify the Service Management Information Entities that are used to configure and schedule CSTSes;

d) The *SLE Transfer Services* suite: The SLE Transfer Services comprise a set of Cross Support Transfer 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. However, because of history (the SLE Transfer Services were already specified and implemented prior to development of the CSTS Framework) the SLE Transfer Services are separated from CSTSes;

e) *Space Link Extension - Internet Protocol for Transfer Services* (reference [G2]): a Recommended Standard that defines a protocol for transfer of Protocol Data Units (PDUs) defined in the Cross Support Transfer Services. This 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:

f) *Cross Support Transfer Services Specification Framework* (reference [1]): a Recommended Standard that defines basic building blocks for the specification of Cross Support Transfer Service procedures;

g) *Guideline for Specification of Cross-Support Transfer Services*: a Recommended Practice that, when published, will define the guidelines for construction of a Cross Support Transfer Service based on the CSTS Specification Framework;

NOTE – As of the publication of this Recommended Standard, the Guidelines Magenta Book is in-progress. It is not required to understand this Recommended Standard.

h) *Cross Support Transfer Services Specification Framework Concepts* (reference [G3]): A Report that provides tutorial material on the objectives and concepts of the CSTS Specification Framework;

i) *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 CSTS Guidelines.

1.7 DEFINITIONS

1.7.1 TERMS DEFINED IN THE CROSS SUPPORT TRANSFER SERVICES SPECIFICATION FRAMEWORK RECOMMENDED STANDARD (REFERENCE [1])

a) Association Control procedure;

b) Buffered Data Delivery procedure;

c) complete data delivery mode;

d) Cross Support Complex;

e) Cross Support Service production;

f) Cross Support Transfer Service;

g) Event Identifier;

h) Event Label;

i) Event Name;

j) Functional Resource instance;

k) Functional Resource Instance Number;

l) Functional Resource Name;

m) Functional Resource Type;

n) Label List;

o) Label List Set;

p) non-blocking (operation);

q) Parameter Identifier;

r) Parameter Label;

s) Parameter Name;

t) prime procedure instance;

u) procedure configuration parameter;

v) procedure type;

w) procedure instance identifier;

x) Published Identifier;

y) qualified parameter;

z) secondary procedure instance;

aa) service management parameter;

bb) service-user-responding-timer;

cc) subscription.

1.7.2 TERMS DEFINED IN THE CROSS SUPPORT REFERENCE MODEL (REFERENCE [2])

a) Complex Management (CM) (called *SLE Complex Management* in reference [2]);

b) Mission User Entity (MUE);

c) service agreement (called *SLE Service Agreement* in reference [2]);

d) service package (called *SLE Service Package* in reference [2]);

e) space link session;

f) transfer service production;

g) transfer service provision;

h) Utilization Management (UM);

i) utilization phase (called SLE Service Package Utilization phase in reference [2]).

1.8 NOMENCLATURE

1.8.1 NORMATIVE TEXT

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

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.

1.8.2 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.

1.9 CONVENTIONS

1.9.1 OVERVIEW

The conventions defined in the CSTS Specification Framework Recommended Standard (reference [1]) are applicable to this Monitored Data service specification, with the exception of the representation of Object Identifiers. The conventions for the representation of Object Identifiers in this Recommended Standard are described in 1.9.2.

1.9.2 OBJECT IDENTIFIER REPRESENTATION

1.9.2.1 General

The MD service involves extensive use of Functional Resource Types, procedure types, Functional Resource Names, procedure instance identifiers, Parameter Names, Parameter Identifiers, Event Names, and Event Labels. As specified in reference [1], all of these names are based on Published Identifiers, which are International Organization for Standardization (ISO) Object Identifiers (OIDs). OIDs have the syntax of strings of integers. For purposes of readability, rather than using actual OIDs in the descriptions and examples in this Recommended Standard, the OIDs in these names and identifiers are represented using the following textual notation to represent the OIDs.

1.9.2.2 Functional Resource Type

As specified in reference [1], a Functional Resource Type is a Published Identifier (i.e., an ISO OID). In the descriptions and examples in this Recommended Standard, a Functional Resource Type is represented using the notation *{published identifier descriptor}*, which is a textual description of the Published Identifier. Thus {Antenna} represents the Published Identifier for the Antenna Functional Resource Type.

1.9.2.3 Procedure Type

As specified in reference [1], a procedure type is an ISO OID that is assigned to a CSTS procedure. In the descriptions and examples in this Recommended Standard, a procedure type is represented using the notation *{object identifier descriptor}*, which is a textual description of the Object Identifier. Thus {Cyclic Report} represents the Object Identifier for the Cyclic Report procedure type.

1.9.2.4 Functional Resource Name

As specified in reference [1], a Functional Resource Name is composed of a Functional Resource Type Published Identifier and an integer Functional Resource Instance Number (FRIN). In the descriptions and examples in this Recommended Standard, a Functional Resource Name is represented using the notation *[{functional resource type published identifier descriptor}: FRIN]*. [{Antenna}: 1] represents the name of the Antenna Functional Resource Type that is assigned FRIN = 1 in the scheduled service package.

1.9.2.5 Procedure Instance Identifier

As specified in reference [1], a procedure instance identifier identifies the specific instance of a procedure in a CSTS. It is composed of a procedure type Published Identifier and a procedure role, which can have one of three values: 'association control' if it is the Association Control procedure of the CSTS; 'prime procedure instance' if it is the prime instance of the procedure of the CSTS; or a positive integer Secondary Procedure Instance Number (SPIN) if it is a secondary procedure instance of the CSTS.

In the descriptions and examples in this Recommended Standard, a procedure instance identifier for an Association Control procedure is represented using the notation *[{procedure type object identifier descriptor}: 'association control']*. [{Association Control}: 'association control'] represents the Procedure Instance Identifier of the Association Control instance of a CSTS.

NOTE – In the case of the Framework Association Control procedure, the Procedure Role appears to be redundant with the procedure type. However, it is possible for a CSTS to extend the Association Control procedure, in which case it would get a separate procedure type, in which case the Procedure Role explicitly identifies the OID as that belonging to an Association Control procedure.

A procedure instance identifier for a prime procedure instance is represented using the notation *[{procedure type object identifier descriptor}: 'prime procedure instance']*. [{Cyclic Report}: 'prime procedure instance'] represents the procedure instance identifier of the Cyclic Report instance that serves as the prime procedure instance of a CSTS.

A procedure instance identifier for a secondary procedure instance is represented using the notation *[{procedure type object identifier descriptor}: SPIN]*. [{Notification}: 1] represents

the procedure instance identifier of the Notification instance that serves as the first Notification secondary procedure instance of a CSTS.

1.9.2.6 Parameter Name

As specified in reference [1], a Parameter Name can be either the name of a Functional Resource parameter or the name of a CSTS procedure parameter. A Functional Resource Parameter Name is composed of (1) the Functional Resource Name of the Functional Resource instance that reports the parameter, and (2) the Parameter Identifier for the parameter. In the descriptions and examples in this Recommended Standard, a Parameter Name is represented using the notation *[[{functional resource type published identifier descriptor}: FRIN]: {parameter published identifier descriptor}]*. Thus [[{Antenna}: 1]: {`actualAzimuth`}] represents the `actual-azimuth` parameter of the Antenna Functional Resource Type that is assigned FRIN = 1 in the scheduled service package.

A CSTS procedure Parameter Name is composed of (1) the procedure instance identifier of the procedure instance that reports the parameter, and (2) the Parameter Identifier for the parameter. In the descriptions and examples in this Recommended Standard, a CSTS Parameter Name is represented using the notation *[[{procedure type object identifier descriptor}: 'association control' | 'prime procedure instance' | SPIN]: {parameter published identifier descriptor}]*. Thus [[{Cyclic Report}: 'prime procedure instance']: {`pCRnamedLabelLists`}] represents the `named-label-lists` parameter of the prime instance of the Cyclic Report procedure of the CSTS.

1.9.2.7 Event Name

As specified in reference [1], an Event Name can be either the name of a Functional Resource event or the name of a CSTS procedure event. A Functional Resource Event Name is composed of (1) the Functional Resource Name of the Functional Resource instance that reports the event, and (2) the Event Identifier for the event. In the descriptions and examples in this Recommended Standard, an Event Name is represented using the notation *[[{functional resource type published identifier descriptor}: FRIN]: {event published identifier descriptor}]*.

NOTE – As of the publication of this Recommended Standard, there are no procedure events for the three procedures used in the Monitored Data service (Cyclic Report, Information Query, and Notification).

1.9.2.8 Parameter Label

As specified in reference [1], a Parameter Label is composed of (1) the Functional Resource Type that reports the parameter and (2) the Parameter Identifier for the parameter. In the descriptions and examples in this Recommended Standard, a Parameter Label is represented using the notation *[{functional resource type published identifier descriptor}: {parameter*

published identifier descriptor}]. Thus [{Antenna}: {`actualAzimuth`}] represents the `actual-azimuth` parameter of the Antenna Functional Resource Type.

1.9.2.9 Event Label

As specified in reference [1], an Event Label is composed of (1) the Functional Resource Type that reports the parameter and (2) the Event Identifier for the event. In the descriptions and examples in this Recommended Standard, an Event Label is represented using the notation *[{functional resource type published identifier descriptor}: {event published identifier descriptor}]*.

1.9.2.10 Label List

As specified in reference [1], a Label List is a data structure that specifies the name of a list of Parameter Labels or Events Labels, indicates if the list is the default list, and contains all Parameter Labels or Event Labels represented by that Label List name. In the descriptions and examples in this Recommended Standard, a Label List is represented using the notation:

{ labelListName;
defaultFlag;
<Parameter Label or Event Label>;
.
.
.
<Parameter Label or Event Label>
}

where *<Parameter Label or Event Label>* is as described in 1.9.2.8 or 1.9.2.9.

1.9.2.11 Label List Set

As specified in reference [1], a Label List Set is the set of Label Lists accessible by the user of the given service instance. In the descriptions and examples in this Recommended Standard, a Label List Set is represented using the notation:

{ <Label List>;
.
.
.
<Label List>
}

where *<Label List>* is as described in 1.9.2.10.

1.10 REFERENCES

[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] "Functional Resources." Space Assigned Numbers Authority. http://sanaregistry.org/r/functional_resources/.

2 OVERVIEW OF THE MONITORED DATA SERVICE

2.1 SERVICE SUMMARY

The Monitored Data CSTS is a CCSDS Cross Support Transfer Service that during the MD-CSTS service instance provision period provides a user with the capability to:

a) obtain cyclic or on-change reports of the values of monitored parameters of interest;

b) obtain notification of the occurrence of events of interest;

c) query the current values of monitored parameters of interest.

An instance of the Monitored Data service progresses through the following activities:

a) the user binds to the provider to establish a service association;

b) the user selects which monitored parameters are to be reported on-change or cyclically and at which reporting periods, and starts (enables) the reporting of those parameters;

c) the user selects which events are to be notified and enables the transmission of those event notifications (optional);

d) the service provider reports the selected monitored parameter values when they change or at the specified periodicity;

e) the service provider transmits the selected notifications upon occurrence of their associated events (if optional event notification (item c), above) has been invoked by the user),

f) the user queries the current values of monitored parameters (optional);

g) the user stops (disables) periodic reporting and event notification (if optional event notification has been invoked); and

h) the user unbinds from the provider to release the association.

The Monitored Data service delivers only current parameter values: there is no capability in this service to deliver parameter values stored from previous times.

2.2 FUNCTIONAL DESCRIPTION

2.2.1 GENERAL

As defined in the Cross Support Reference Model: Part 1 (reference [2]), related Cross Support Services are bundled into service packages for the purposes of ensuring that the required relationships among those Cross Support Services are preserved during their production and provision. For example, multiple Cross Support Transfer Services may be related to the operation of the same radio frequency (RF) link, and the return RF link may be related to the forward RF link; all of those transfer services, as well as the RF links themselves, constitute a single service package for the purposes of scheduling. The content and structure of service packages that are compatible with the MD-CSTS will be defined in a future Service Management Recommended Standard (reference [G6]).

A service package may contain multiple instances of the Monitored Data service. Each instance of the Monitored Data service in a service package is capable of reporting all monitored parameters and event notifications for that service package. The example service package used in this section is comprising a forward RF link, two return RF links, multiple SLE transfer service instances, and an instance of the Monitored Data service. The Monitored Data service instance has access to the monitored parameters and event notifications for those RF links and SLE transfer service instances (in addition to the Monitored Data service instance's own monitored parameters and event notifications).

In accordance with reference [2], the functionality associated with a transfer service is partitioned into production and provision of the service. The following subsections describe the production and provision of the Monitored Data service, respectively.

2.2.2 SERVICE PRODUCTION

The production of the Monitored Data service associated with a given service package consists of:

- various space communication and radiometric functions that constitute that service package; and
- a Monitored Data Collection function that collects the monitored parameter measurements and event notifications emitted by those functions in order to make them available to the Monitored Data service instances.

Figure 2-1 is a notional representation of the relationship between the functions of a service package and instances of the Monitored Data service that report on that service package.

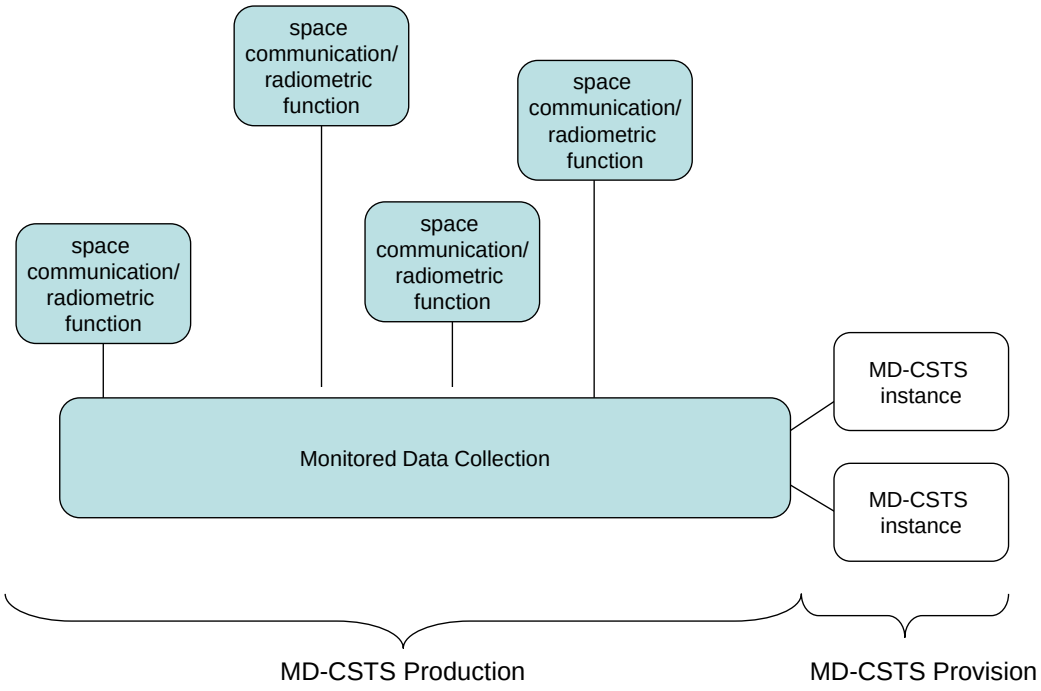

Figure 2-1: Production and Provision of Monitored Data Services (Notional)

In order for the Monitored Data service to represent the monitored parameter values and event notifications of the space communication and radiometric functions in a consistent manner, these functions are represented as standardized *Functional Resources*.

As defined in reference [1], a *Functional Resource Type* is a logical function or related set of functions that provides a space communication capability. A Functional Resource Type may have configuration parameters, monitored parameters, notifiable events, and/or directives associated with it. For the purposes of the Monitored Data CSTS, only the monitored parameters and notifiable events of Functional Resource Types are of interest.

In general, a Functional Resource Type may be instantiated multiple times in the same service package, and therefore it is necessary to distinguish the individual instances of those Functional Resource Types, known as *Functional Resource instances*.

A standard common set of space communication Functional Resource Types and their associated parameters (both configuration and monitored), notifiable events, and directives have been defined by CCSDS and registered with SANA in accordance with the process specified in reference [1]. Annex F identifies an example set of Functional Resource Types and the monitored parameters and notifiable events associated with those Functional Resource Types. These example types, parameters, and events are consistent with the SANA Functional Resource Registry (reference [3]) at the time of publication of this Recommended Standard, but may change over time without annex F being updated. Annex F should therefore be considered informative, and implementers of the MD-CSTS must use the SANA

Functional Resource Registry as the normative source for definitions of the applicable Functional Resource Types, parameters, and events.

NOTES

1 The operational scenario in 2.5 provides hypothetical examples of Functional Resource Types.

2 Cross Support Complexes that provide the MD-CSTS will be expected to report as many of the standard monitored parameters and notifiable events as possible. Individual Cross Support Complexes may also use the MD-CSTS to report non-standard monitored parameters and event notifications that are available from those Complexes.

The execution of a typical service package is realized through the operation of instances of different types of Functional Resources (e.g., Forward 401 Space Link Carrier Transmission, Return 401 Space Link Carrier Reception) and even multiple instances of the same Functional Resource Type(s) (e.g., a Return Telemetry Synchronization and Channel Decoding Functional Resource instance associated with an X-Band return space link and an instance of that Functional Resource Type associated with an S-Band return space link). Conceptually, instrumentation of the Functional Resources produces the raw measurements that are the basis for the monitored parameter values and the event notifications associated with those Functional Resources.

The MD-CSTS is not constrained to operate with any particular set of Functional Resource Types, but rather can report the monitored parameter values and event notifications for any set of Functional Resources as long as the monitored data and event notifications emitted by those Functional Resources are defined in the SANA Functional Resource Registry.

Functional Resources may occur in different combinations and multiplicities within a service package. For example, a single antenna may be used by both forward and return space links, and more than one return space link may use the same antenna. If Quadrature Phase Shift Keying (QPSK) modulation is used on the carrier, there may be two symbol streams multiplexed onto the single carrier. It is therefore possible that a single service package may have multiple instances of the same types of monitored parameters or notifiable events associated with the different instances of the Functional Resources (e.g., bit synchronizer lock status for a service package that encompasses multiple return symbol streams).

To provide unambiguous identification of each instance of a monitored parameter or notifiable event within the context of the service package, the MD-CSTS uses the Parameter Name of each monitored parameter that is reported and the Event Name of each event that is notified. The Parameter Name of a parameter represents both (a) the type of monitored parameter (e.g., `forward-401-carrier-transmission-actual-carrier-frequency`) and (b) the name of the Functional Resource instance with which the parameter is associated. Similarly, the Event Name of a notifiable event represents both (a) the type of the notifiable event (e.g., ‘`return-telemetry-synchronization-`

`frame-synchronizer-lock-status-change`') and (b) the name of the Functional Resource instance with which the event is associated.

As specified in the CSTS Specification Framework Recommended Standard (reference [1]):

a) the monitored parameter type component of the Parameter Name is called the Parameter Identifier and has the syntax of a Published Identifier, that is, a special type of Object Identifier on the CCSDS Registration Tree;

b) the notifiable event type component of the Event Name is called the Event Identifier and has the syntax of a Published Identifier;

c) the name of the Functional Resource instance has the syntax of a Functional Resource Name which has two components:

   1) a Functional Resource Type, which has the syntax of a Published Identifier, and

   2) a Functional Resource Instance Number, which is a positive integer.

NOTES

1 The Functional Resource Instance Number is used to differentiate between multiple instances of the same Functional Resource Type that are configured in the same service package. Whereas the Parameter Identifiers, Event Identifiers, and Functional Resource Types are all statically defined, the assignment of Functional Resource Instance Numbers is a dynamic result of the scheduling of service packages, which may contain different combinations and multiplicities of Functional Resource Types.

2 The abstract syntaxes of the Parameter Name, Event Name, Functional Resource Type, Functional Resource Instance Number, and Functional Resource Name are formally specified in reference [1].

The Monitored Data Collection function of MD-CSTS production makes all of the monitored parameters and event notifications available to all instances of MD-CSTS of the service package in conformance with the naming, syntax, type, and units defined in the specification for those monitored parameters and event notifications. In cases where the real implementations of Functional Resources do not present their monitored data measurements and/or event notifications in conformance with the naming, syntax, and/or units required for transfer via the MD-CSTS, the Data Collection function performs the appropriate format conversion.

During the execution of a service package, the production collectively supplies two kinds of data for use in the provision of MD-CSTS instances:

a) the current values of all instances of functional resource monitored parameters that are collected and made available by the Complex for either cyclic or on-change reporting to, or query by, that spaceflight mission; and

b) functional resource event notifications for all notifiable events that are made available by the Complex for that spaceflight mission.

2.2.3 SERVICE PROVISION

2.2.3.1 General

Multiple instances of the MD-CSTS may exist in the same service package, with each instance providing selected subsets of the functional resource monitored parameter values and event notifications that have been aggregated from the monitored Functional Resources.

Each instance of the Monitored Data service allows the user to subscribe to any of the sets of monitored parameters supported by the Complex and have that set of parameters be cyclically or on-change reported via that service instance. Optionally, each instance of the Monitored Data service allows the user to query the current value of any monitored parameter. Also optionally, each instance of the Monitored Data service allows the user to subscribe to any of the supported set of notifiable events to have those specific events to be notified via that MD-CSTS instance.

NOTE – The capabilities of real implementations of Functional Resources or the Data Collection function of the production affect the freshness and/or accuracy of the data being reported. For example, a particular piece of equipment may output a new measurement value only every 5 seconds. An implementation of the data collection function might hold this value as a constant in the database for the duration of the 5 seconds. In this case, a resulting query that is invoked half-way between samples would return the 2.5-second-old value as the ‘current’ value. Agreements between spaceflight mission and Cross Support Complexes should specify any such timing relationships.

2.2.3.2 Cyclic Reporting of Parameters

The MD-CSTS On-Change-Option Cyclic Report procedure (see section 4) allows the service user to subscribe to a *list of parameters* to be periodically reported, specify the *delivery cycle* (period) for those reports, and specify whether every subscribed parameter value is to be reported at every delivery cycle or only when the parameter value has changed since the previous report. These delivery selections are specified in the START invocation message that activates the On-Change-Option Cyclic Report procedure. Multiple instances of the On-Change-Option Cyclic Report procedure can be active concurrently, each with a different set of subscribed parameters, delivery cycle, and on-change designation, which allows the service user to fine tune the reporting based on relative importance and expected rate of change in the values of the monitored parameters.

The On-Change-Option Cyclic Report procedure is the prime procedure for the MD service. That is, at least one instance of the On-Change-Option Cyclic Report procedure must be active in order for the MD service itself to be considered active.

Once an On-Change-Option Cyclic Report procedure instance is activated, it reports the values of the subscribed parameters at the specified delivery cycle until a subsequent STOP invocation message for that procedure instance is received.

The list of parameters in the START invocation of the On-Change-Option Cyclic Report procedure may contain:

a) the names of individual monitored parameters that the Cross Support Complex has previously agreed to support, for all instances of all Functional Resources in the service package;

b) the labels of monitored parameter types;

NOTE – Labels provide a form of wildcard selection—when a Parameter Label is used, the values of all instances of that parameter type are reported for all instances of all Functional Resources in the service package.

c) the name of a list of Parameter Labels that has been agreed between the Cross Support Complex and the Mission;

NOTE – Such lists can be used to bundle monitored parameters that are routinely used as a group by the service user. A special case of the Parameter Label List is the *default list*, which is a Parameter Label List that is used by default when MD-CSTS user leaves the list of parameters empty;

d) the name of a specific Functional Resource instance, which is another wildcard selection mechanism, this one causing all parameter values for the named Functional Resource instance to be reported; or

e) the identifier of a Functional Resource Type, yet another wildcard selection mechanism, this one causing all parameter values for all instances of that Functional Resource Type that occur in the service package to be reported.

Optionally (by implementation), the list of parameters may alternatively contain:

a) the names of procedure configuration parameters, in order to receive periodic reports of the values of the configuration parameters of the procedures that constitute the MD service;

b) the identifier of a procedure type, a wildcard selection mechanism that causes the values of all configuration parameters of all instances of the specified procedure type to be reported; or

c) a procedure instance identifier, a wildcard selection mechanism that causes the values of all configuration parameters of the specified procedure instance to be reported.

NOTE – The configuration parameters of the procedures that constitute the MD service are all static (that is, they do not change during the execution of the MD service). Some of these configuration parameters may be voluminous and cyclic reporting may have adverse operational effects. Therefore cyclic reporting of procedure configuration parameters is optional.

If the list of parameters in the START invocation contains any Parameter Names, Parameter Labels, list names, Functional Resource Types, Functional Resource Names, procedure types, or procedure instance identifiers that are unknown to or not supported by the Cross Support Complex, the START operation fails and a list of the unknown Parameter Names, Labels, etc., is returned to the service user.

At least one instance of the On-Change-Option Cyclic Report procedure is mandatory for all implementations, but each implementation sets the maximum number of instances that can be active during the service instance provision period of the MD-CSTS.

2.2.3.3 Event Notifications

The MD-CSTS Notification procedure (see section 6) allows the service user to subscribe to a *list of events* in the START invocation message that activates the Notification procedure. Once a Notification procedure instance is activated, it will notify the service user of any of the subscribed events that occur between then and the time that a subsequent STOP invocation message for that procedure instance is received.

The START invocation may subscribe to notifiable events by using Events Names, Event Labels, an Event Label List (named or default), a Functional Resource Name, or a Functional Resource Type.

The Notification procedure is optional for MD-CSTS implementations. If an implementation supports the Notification procedure, multiple instances of the Notification procedure can be active concurrently, with each implementation setting the maximum number of instances that can be active during the service instance provision period of the MD-CSTS.

2.2.3.4 Query of Parameters

The Information Query procedure (see section 5) allows the service user to query the current values of functional resource monitored parameters and MD-CSTS procedure configuration parameters. The Information Query procedure consists of a single GET operation that can be used to query the values of different parameters every time that it is invoked.

The GET invocation message can be used to query functional resource parameter values using Functional Resource Parameter Names, Parameter Labels, a Parameter Label List (named or default), a Functional Resource Name, or a Functional Resource Type.

The GET invocation message can also use procedure configuration Parameter Names, a procedure type, or a procedure instance identifier to query the values of configuration parameter of the procedures that constitute the MD-CSTS instance.

The Information Query procedure is optional for MD-CSTS implementations. Because a single instance of the Information Query procedure can be used to query any of the queriable parameters of the MD-CSTS, the MD-CSTS supports at most one instance of the Information Query procedure during the service instance provision period.

NOTE – The GET operation can be invoked any number of times during the service instance provision period. Furthermore, because the GET operation of the Information Query procedure is non-blocking (as defined in reference [1]), the GET operation can be invoked before the responses to previous invocations are returned.

2.3 SERVICE MANAGEMENT

As defined in the Cross Support Reference Model: Part 1 (reference [2]), related Cross Support Services are bundled into service packages for the purposes of scheduling. Cross Support Service Management both establishes the constraints on the service packages to which a given spaceflight mission must conform (e.g., data rate and frequency ranges, types and numbers of Cross Support Transfer Service instances) and provides the mechanisms for instantiating conformant service packages (e.g., via scheduling).

As described in 2.2, the Monitored Data service uses the Functional Resource representation of the space communication and radiometric functions to report monitored parameter values and event notifications. Functional Resources are also used by Service Management to represent these functions. That is, service packages are expressed in terms of Functional Resource instances, and the initial values of the configuration parameters of those service packages are expressed in terms of the standardized configuration parameters of the respective Functional Resource Types.

With regard to the production and provision of Monitored Data service instances:

a) Cross Support Service Management may establish one or more named lists of Parameter Labels to be used to request groups of monitored parameters to be cyclically reported (and queried if the optional query capability is supported) during the MD-CSTS service instance provision period. One of these lists may be designated the default list of Parameter Labels, which means that it will be applied unless Parameter Names, Parameters Labels, etc., are explicitly requested. As specified in reference [1], each Parameter Label is composed of (1) the Parameter Identifier of that parameter and (2) the Functional Resource Type with which that Parameter Identifier is associated.

b) Cross Support Service Management may (if the Notification procedure is supported by the Complex) establish one or more named lists of Event Labels to be used to request groups of notifiable events to be reported during the MD-CSTS service

instance provision period. One of these lists may be designated the default list of Event Labels, which means that it will be applied unless Event Names, Event Labels, etc., are explicitly requested. As specified in reference [1], each Event Label is composed of (1) the Event Identifier of that event and (2) the Functional Resource Type with which that Event Identifier is associated.

c) Cross Support Service Management schedules the service packages which specify both the Cross Support Services that are to be monitored and the Monitored Data service instances that transfer the monitored parameters to the users of the Monitored Data service instances.

The means by which Service Management performs these functions is outside the scope of this Recommended Standard.

2.4 CROSS SUPPORT VIEW

Figure 2-2 shows an example configuration of a Cross Support Complex providing instances of Monitored Data service to a Mission Data Operations System (MDOS). In this example, the Functional Resources that are configured to support the service package (and therefore available for monitoring via the MD-CSTS) are the Space Link Extension (SLE) Forward Communications Link Transmission Unit (Forward CLTU) transfer service (reference [G4]), the Functional Resource instances that are associated with F-CLTU production (Antenna, Forward 401 Space Link Carrier Transmission, and Forward TC PLOP, Sync, and Channel Encoding), the SLE Return All Frames transfer service (reference [G5]), and the Functional Resource instances that are associated with Return All Frames production (Antenna, Return 401 Space Link Carrier Reception, and Return Telemetry Synchronization and Channel Decoding).

NOTES

1 ‘401 Space Link’ in the string identifier of the Functional Resource Types indicates that these types are suitable only to model space links that conform to CCSDS 401.0-B-26 (reference [G10]) but not for space links that conform, e.g., to CCSDS 415.1-B-1 (reference [G11]).

2 A given Functional Resource Type may represent several functions of which, depending on the type of input data, only a subset is applicable. The reason for this is twofold: (a) this approach limits the number of Functional Resource Types to a manageable number and (b) permits, where feasible, grouping the functions such that those of one Functional Resource Type correspond to one CCSDS Recommended Standard. The ‘Forward TC PLOP, Sync and Channel Encoding’ Functional Resource Type is one example of this approach. In terms of functions it corresponds to the ‘TC Synchronization and Channel Coding’ book (reference [G12]). It should be noted that, although not obvious from the title of that Recommended Standard, it not only specifies the synchronization mechanism required to delimit CLTUs from each

other by means of start and tail sequences, but also the Physical Layer Operations Procedure (PLOP) that permits the spacecraft to achieve carrier and symbol lock.

3 In the scenario illustrated in figure 2-2, the input data are readily configured CLTUs including start and tail sequences. The only thing missing in terms of generating the symbol stream that shall modulate the forward carrier is the execution of the Physical Layer Operations Procedure and that is in the depicted scenario the only function of the 'Forward TC PLOP, Sync and Channel Encoding' Functional Resource instance being used. In a different scenario the input data can be TC frames and in that case the coding and synchronization functions would be used in the process of converting incoming TC frames to CLTUs.

The SLE transfer service instances and the production Functional Resources associated with them are not part of the Monitored Data service, but they are monitored through the Monitored data service.

NOTE – For the purposes of this description, the production Functional Resources named above and shown in the figure are generic versions of the Functional Resource Types registered with SANA at the time of publication of this Recommended Standard. The actual names of the Functional Resources are as specified in the SANA Functional Resource Registry (reference [3]).

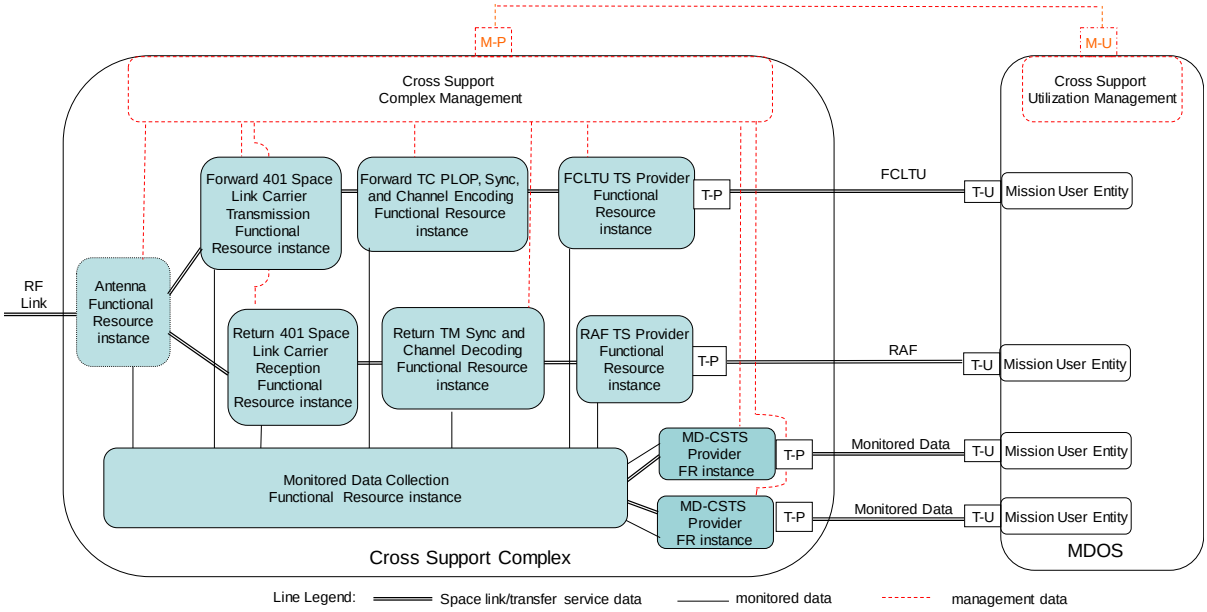

Figure 2-2: Example of the Management and Provision of Monitored Data Service Instances for a Service Package

As shown in figure 2-2, a single service package may include multiple instances of Monitored Data service. All Monitored Data service instances (i.e., all MD-CSTS users) have access to all monitored parameters and notifiable events associated with the service package.

As also shown in the figure, the MD-CSTS instances and the Monitored Data Collection function are themselves monitored Functional Resources. Any service management parameters or notifiable events associated with the MD-CSTS instances are made available via those MD-CSTS instances in the same way that such information about any Functional Resource instance (including Monitored Data Collection) is made available.

NOTE – According to the Cross Support Reference Model (reference [2]), the responsibility for monitoring the overall state of the execution of a service package is associated with the Utilization Management (UM) role of the MDOS, which is a Service Management role. In this case, a Service Management *function* is being performed, but it is being performed using a CSTS rather than using a management service built upon the Space Communication Cross Support Service Management infrastructure (references [G6], [G7], and [G8]). Although figure 2-2 shows the Mission User Entity (MUE) for the MD-CSTS as being separate from the UM role, it should be understood that in the nominal case the use of the MD-CSTS will be in support of a Service Management responsibility. However, there may be other uses of monitored data that are not linked to UM. For example, the principal investigator for a particular instrument may need to correlate his or her instrument data (returned via the Return All Frames SLE service) with communications service status that is not available via the Return All Frames service. Such a principal investigator could be a user of an MD-CSTS instance and subscribe to the monitored parameters of interest.

2.5 OPERATIONAL SCENARIO

2.5.1 GENERAL

This subsection presents an operational scenario of the MD service. This scenario is written using the Functional Resource Name, Parameter Name, Parameter Label, Event Name, Event Label, procedure type, and procedure instance identifier representation conventions described in 1.9.2.

NOTES

1 The representation conventions in 1.9.2 represent the OID components as text descriptors enclosed in curly braces; e.g., {Association Control} represents the OID assigned as the procedure identifier for the Association Control procedure. The OIDs associated with Functional Resources (Functional Resource Types, Functional Resource Parameter Identifiers, and Functional Resource Event Identifiers) are formally registered with SANA.

2 Annex F identifies the subset of Functional Resources that are used in this operational scenario. For each Functional Resource Type, annex F identifies the OID for that Functional Resource Type, the monitored parameters and notifiable events that are used in this operational scenario. These Functional Resource definitions, parameters, notifiable events, and respective OID assignments are based on the candidate SANA

Functional Resource Registry that existed as of the time of publication of this Recommended Standard. The normative definitions that are recorded in the SANA Functional Resource Registry at any given time may differ from those in annex F.

3 The OIDs associated with CSTS Framework procedures (procedure types, procedure configuration Parameter Identifiers, and procedure Event Identifiers) are specified in the CSTS Specification Framework Recommended Standard (reference [1]).

The following scenario is success-oriented. That is, no error conditions are addressed.

2.5.2 SERVICE AGREEMENT/SERVICE PLANNING ACTIVITIES

As part of the Service Management activities that establish the relationship between the Xenosat mission and the Multinet Cross Support Complex, UM for Xenosat and Complex Management (CM) for Multinet negotiate the set of Cross Support Services that will be available to the Xenosat mission within the context of the service agreement.

For the purpose of this scenario, the Cross Support Services include a single S-Band forward space link and associated single instance of Forward CLTU SLE transfer service (reference [G4]), an S-Band return space link and associated single instance of Return All Frames SLE transfer service (reference [G5]), and an X-Band return space link and associated single instance of Return All Frames transfer service. Multinet supports both the optional Notification and Information Query procedures of the MD-CSTS standard in addition to the mandatory On-Change-Option Cyclic Report procedure. The service agreement also includes a single instance of the MD-CSTS, which supports four instances of each of the On-Change-Option Cyclic Report and Notification procedures, and one instance of the Information Query procedure.

The Functional Resource instances associated with these Cross Support Services are:

a) two instances of the Antenna Functional Resource Type (see F2), one for X-Band and one for S-Band, with Functional Resource Name ‘[{`antenna`}: 1]’ for the X-Band Antenna and Functional Resource Name ‘[{`antenna`}: 2]’ for the S-Band Antenna;

b) one instance of the Forward 401 Space Link Carrier Transmission Functional Resource Type (see F3), with Functional Resource Name ‘[{`fwd401SpaceLinkCarrierXmit`}: 1]’ for the Forward S-Band;

c) one instance of the Forward TC PLOP, Sync, and Channel Encoding Functional Resource Type (see F4) for the Forward S-Band link, with Functional Resource Name ‘[{`fwdTcPlopSyncAndChnlEncoding`}: 1]’;

d) one instance of the Forward CLTU Transfer Service Provider Functional Resource Type (see F5), with Functional Resource Name ‘[{`fwdCltuTsProvider`}: 1]’;

e) two instances of the Return 401 Space Link Carrier Reception Functional Resource Type (see F6), one for X-Band and one for S-Band, with Functional Resource Name

‘[{rtn401SpaceLinkCarrierRecpt}: 1]’ for the X-Band Return 401 Space Link Carrier Reception Functional Resource instance and Functional Resource Name ‘[{rtn401SpaceLinkCarrierRecpt}: 2]’ for the S-Band Return 401 Space Link Carrier Reception Functional Resource instance;

NOTE – In this example the data are modulated onto subcarriers that are Bi-Phase Shift Keying (BPSK)-modulated, so that only one symbol stream is carried by each subcarrier.

f) two instances of the Return TM Synchronization and Channel Decoding Functional Resource Type (see F7), one for the X-Band return space link symbol stream and one for the S-Band return space link symbol stream, with Functional Resource Name ‘[{rtnTmSyncAndChnlDecoding}: 1]’ for the X-Band Return TM Synchronization and Channel Decoding Functional Resource instance and Functional Resource Name ‘[{rtnTmSyncAndChnlDecoding}: 2]’ for the S-Band Return TM Synchronization and Channel Decoding Functional Resource instance;

g) two instances of the Return All Frames Transfer Service Provider Functional Resource Type (see F8), one for the X-Band return space link symbol stream and one for the S-Band return space link symbol stream, with Functional Resource Name ‘[{rtnAfTsProvider}: 1]’ for the X-Band Return All Frames Transfer Service Provider Functional Resource instance and Functional Resource Name ‘[{rtnAfTsProvider}: 2]’ for the S-Band Return All Frames Transfer Service Provider Functional Resource instance;

h) one instance of the Monitored Data CSTS Provider Functional Resource Type (see F9) for the service package, with Functional Resource Name ‘[{mdCstsProvider}: 1]’ for the MD-CSTS Provider Functional Resource instance;

i) one instance of the Monitored Data Collection Functional Resource Type (see F10) for the service package, with Functional Resource Name ‘[{mdCollection}: 1]’ for the Monitored Data Collection Functional Resource instance.

Among the Functional Resource monitored parameters collected and made available by the Multinet Complex are:

– the antPointingMode for the Antenna Functional Resource Type, with Parameter Label [{antenna}: {antPointingMode}] (see F2 for the OID values);

– the fwd401CarrierXmitActualCarrierFreq for the Forward 401 Space Link Carrier Transmission FR Type, with Parameter Label [{fwd401SpaceLinkCarrierXmit}: {fwd401CarrierXmitActualCarrierFreq}] (see F3 for the OID values);

– the fwdCltuTsSiState (Forward CLTU Transfer Service Service Instance State) and fwdCltuTsNumberOfCltusRadiated for the Forward CLTU Transfer Service Provider FR Type, with Parameter Labels [{fwdCltuTsProvider}:

{fwdCltuTsSiState}] and [{fwdCltuTsProvider}: {fwdCltuTsNumberOfCltusRadiated}], respectively (see F5 for the OID values);

– the rtn401CarrierRecptActualFreq, rtn401CarrierRecptSubcarrierLockStat, rtn401CarrierRecptActualSubcarrierFreq, and rtn401CarrierRecptSymbolSyncLockStat for the Return 401 Space Link Carrier Reception FR Type, with Parameter Labels [{rtn401SpaceLinkCarrierRecpt}: {rtn401CarrierRecptActualFreq}], [{rtn401SpaceLinkCarrierRecpt}: {rtn401CarrierRecptSubcarrierLockStat}], [{rtn401SpaceLinkCarrierRecpt}: {rtn401CarrierRecptActualSubcarrierFreq}], and [{rtn401SpaceLinkCarrierRecpt}: {rtn401CarrierRecptSymbolSyncLockStat}] (see F6 for the OID values); and

– the rtnAfSiState and the rtnAfNumberOfFramesDelivered for the Return All Frames Transfer Service Provider FR Type, with Parameter Labels [{rtnAfTsProvider}: {rtnAfSiState}] and [{rtnAfTsProvider}: {rtnAfNumberOfFramesDelivered}], respectively (see F8 for the OID values).

Among the notifiable events made available by the Multinet Complex are:

– rtnTmSyncFrameSyncLockStatChange for the Return TM Synchronization and Channel Decoding FR Type, with Parameter Label [{rtnTmSyncAndChnlDecoding}: {rtnTmSyncFrameSyncLockStatChange}] (see F7 for the OID values);

– mdCstsProdStatChange for the MD CSTS Provider FR Type, with Event Label [{mdCstsProvider}: {mdCstsProdStatChange}] (see F9 for the OID values);

– rtnAfProdStatChange for the Return All Frames Transfer Service (TS) Provider FR Type, with Parameter Label [{rtnAfTsProvider}: {rtnAfProdStatChange}] (see F8 for the OID values); and

– fwdCltuTsProdStatChange for the Forward CLTU TS Provider FR Type, with Parameter Label [{fwdCltuTsProvider}: {fwdCltuTsProdStatChange}] (see F5 for the OID values).

As part of the Service Management activities that establish the relationship between the mission and the Complex, UM and CM negotiate a single monitored Parameter Label List that will serve as the default Label List for both the On-Change-Option Cyclic Report and

Information Query procedures. In this scenario, the default Label List is given the name ‘defaultLabelList’ and contains the Parameter Labels for the Forward CLTU TS Provider `fwdCltuTsNumberOfCltusRadiated` parameter and the Return All Frames TS Provider `rtnAfNumberOfFramesDelivered` parameter. The `defaultList` parameter of the Label List is set to TRUE to indicate that it is the default Label List.

```
{   "defaultLabelList";
    TRUE;
    [{fwdCltuTsProvider }: {fwdCltuTsNumberOfCltusRadiated}];
    [{rtnAfTsProvider }: {rtnAfNumberOfFramesDelivered}]
}
```

The default Label List (and any other Label Lists that might be created) can be used by any On-Change-Option Cyclic Report or Information Query procedure instance of any MD-CSTS instance in any service package that is established in the context of the service agreement.

Prior to the time at which cross support is desired, Utilization Management causes Complex Management to create a service package that specifies a space link session with the S-Band forward link and both S-Band and X-Band return links to be provided and the start and stop times for each of the associated service provisions and productions. Included in the service package is an instance of a Monitored Data Cross Support Transfer Service.

2.5.3 SERVICE PACKAGE EXECUTION

2.5.3.1 Binding the MD-CSTS Instance

As of the scheduled beginning of the service instance provision period of the Monitored Data service instance, the service instance exists in the ‘unbound’ state. At the scheduled start time of the space link services and the production of the associated transfer services, the Complex establishes the space links with the spacecraft and begins processing of the signals to and from the spacecraft. Any time after the scheduled start of the service instance provision periods of the Return All Frames and Forward CLTU service instances, the users of those services may bind to those services and use them.

At any time following scheduled beginning of the service instance provision period of the Monitored Data service, the user of that service invokes the BIND operation of the Association Control procedure instance to bind to the service provider, transition the service instance to the ‘bound.ready’ state, and place each of the other procedure instances in the ‘inactive’ state. In this scenario, the scheduled beginning of the service instance provision period of the Monitored Data service is prior to the time at which the associated production process becomes operational, and the binding to the service instance occurs while the production status of the service is still merely configured.

2.5.3.2 Prime Instance of On-Change-Option Cyclic Report Procedure

Following the successful binding of the service instance, the user of the Monitored Data service invokes the START operation for the prime instance of the On-Change-Option Cyclic Report Procedure, which places the prime procedure in the ‘active’ state and the service instance in the ‘bound.active’ state. The parameter of the START invocation that is used to subscribe to monitored parameters is empty, thus indicating that the parameters of the default Label List are to be reported. The START invocation also specifies that the reporting is to occur at two-second intervals.

The MD-CSTS provider invokes the TRANSFER-DATA operation of the prime On-Change-Option Cyclic Report procedure instance to report the values of the monitored parameters at the specified two-second-interval. Each invocation of the TRANSFER-DATA operation for this instance of the On-Change-Option Cyclic Report procedure transfers the values for the monitored parameters with the following Parameter Names:

– [[{`fwdCltuTsProvider`}: 1]: {`fwdCltuTsNumberOfCltusRadiated`}];

– [[{`rtnAfTsProvider`}: 1]: {`rtnAfNumberOfFramesDelivered`}]; and

– [[{`rtnAfTsProvider` }: 2)]: {`rtnAfNumberOfFramesDelivered`}].

The use of a Label List causes every instance of a parameter with a Parameter Label that is in the list to be reported. Thus `rtnAfNumberOfFramesDelivered` is reported for both instances of the Return All Frames TS Provider Functional Resource because both have the same Parameter Label.

Because the prime instance of the On-Change-Option Cyclic Report Procedure is started when the production of the MD CSTS is simply ‘configured’ and not yet ‘operational’ values for the three parameters are reported as ‘unavailable’ until the MD service production becomes operational, at which time the actual values of the parameters begin to be reported.

2.5.3.3 Second Instance of On-Change-Option Cyclic Report Procedure

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS starts the second On-Change-Option Cyclic Report Procedure instance to cyclically report a group of parameters every ten seconds. The START invocation lists the following Parameter Names:

– [[{`fwd401SpaceLinkCarrierXmit`}: 1]: {`fwd401CarrierXmitActualCarrierFreq`}];

– [[{`fwdCltuTsProvider`}: 1]: {`fwdCltuTsSiState`}];

– [[{`rtnAfTsProvider`}: 1]: {`rtnAfSiState`}]; and

– [[{`rtnAfTsProvider`}: 2]: {`rtnAfSiState`}].

The MD-CSTS provider invokes the TRANSFER-DATA operation of the second On-Change-Option Cyclic Report procedure instance to report the values of the monitored parameters at the specified ten-second-interval. Each invocation of the TRANSFER-DATA operation for this instance of the On-Change-Option Cyclic Report procedure transfers the values for the monitored parameters with the following Parameter Names:

– [[{`fwd401SpaceLinkCarrierXmit`}: 1]: {`fwd401CarrierXmitActualCarrierFreq`}];

– [[{`fwdCltuTsProvider`}: 1]: {`fwdCltuTsSiState`}];

– [[{`rtnAfTsProvider`}: 1]: {`rtnAfSiState`}]; and

– [[{`rtnAfTsProvider`}: 2]: {`rtnAfSiState`}].

As with the prime procedure instance, the values for these parameters are reported as ‘unavailable’ while the production status of the Monitored Data service is ‘configured’.

NOTE – Any secondary instance of the On-Change-Option Cyclic Report procedure may be started at any time that the MD-CSTS instance is in the ‘bound’ state. In particular, any secondary instance of the On-Change-Option Cyclic Report procedure may be started regardless of the state of the On-Change-Option Cyclic Report prime procedure instance. That is, any secondary instance of the On-Change-Option Cyclic Report procedure may be active when the MD-CSTS instance is in either the ‘bound.ready’ or ‘bound.active’ state.

2.5.3.4 Third Instance of On-Change-Option Cyclic Report Procedure

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS starts the third On-Change-Option Cyclic Report Procedure instance to cyclically report (at ten-second intervals) all parameters of the S-Band Return Space Link Carrier Reception Functional Resource by using Functional Resource Name in the list-of-parameters:

– [{`rtn401SpaceLinkCarrierRecpt`}: 2].

The MD-CSTS provider invokes the TRANSFER-DATA operation of the third On-Change-Option Cyclic Report procedure instance to report the values of the monitored parameters at the specified ten-second-interval. Each invocation of the TRANSFER-DATA operation for this instance of the On-Change-Option Cyclic Report procedure transfers the values for the monitored parameters with the following Parameter Names:

– [[{`rtn401SpaceLinkCarrierRecpt`}: 2]: {`rtn401CarrierRecptSubcarrierLockStat`}];

– [[{`rtn401SpaceLinkCarrierRecpt`}: 2]: {`rtn401CarrierRecptActualSubcarrierFreq`}].

The values for these parameters are reported as ‘unavailable’ while the production status of the Monitored Data service is ‘configured’.

2.5.3.5 First Instance of Notification Procedure

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS starts the first Notification procedure instance to subscribe to the following Event Label on occurrences of changes in the state of the frame synchronizer:

– [{`rtnTmSyncAndChnlDecoding`}: {`rtnTmSyncFrameSyncLockStatChange`}].

By using Event Label instead of Event Name in the subscription, the user subscribes to the `rtnTmSyncFrameSyncLockStatChange` event notification for all instances of Return TM Synchronization and Channel Decoding that are configured as part of the service package.

When the return S-Band frame synchronizer locks, the MD-CSTS provider invokes the NOTIFY operation of the Notification procedure instance to signal the occurrence of the event with the Event Name

[[{`rtnTmSyncChnlDecoding`}: 2]: {`rtnTmSyncFrameSyncLockStatChange`}].

This event carries the current value (‘locked’) for the frame synchronizer lock status on the return S-Band link.

When the return X-Band frame synchronizer locks, the MD-CSTS provider invokes the NOTIFY operation of the Notification procedure instance to signal the occurrence of the event with the Event Name

[[{`rtnTmSyncChnlDecoding`}: 1]: {`rtnTmSyncFrameSyncLockStatChange`].

This event carries the current value (‘locked’) for the frame synchronizer lock status on the return X-Band link.

NOTE – Any instance of the Notification procedure may be started at any time that the MD-CSTS instance is in the ‘bound’ state. In particular, any instance of the Notification procedure may be started regardless of the state of the On-Change-Option Cyclic Report prime procedure instance. That is, any instance of the Notification procedure may be active when the MD-CSTS instance is in either the ‘bound.ready’ or ‘bound.active’ state.

2.5.3.6 Second Instance of Notification Procedure

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS starts the second Notification procedure instance to subscribe to the notifiable events of the following

Functional Resource Type on occurrences of changes in the production status of the production processes for the MD CSTS Provider:

{mdCstsProvider}.

By using Functional Resource Type in the subscription, the user subscribes to all event notifications for the MD CSTS Provider instance that is configured as part of the service package. This includes all production status-related notifications for the instance of that Functional Resource Type.

When the Monitored Data production status transitions from ‘configured’ to ‘operational’, the MD-CSTS provider invokes the NOTIFY operation of the Notification procedure instance to signal the occurrence of the event with the Event Name

[[{mdCstsProvider}: 1]: {mdCstsProdStatChange}].

This event carries the current value (‘operational’) of the production status of the MD-CSTS Provider.

2.5.3.7 Third Instance of Notification Procedure

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS starts the third Notification procedure instance to subscribe to the notifiable events of the following Functional Resource Type on occurrences of changes in the production status of the production processes for the Return All Frames TS Providers:

{rtnAfTsProvider}.

By using the Functional Resource Type in the subscription, the user subscribes to all event notifications for all instances of the Return All Frames TS Provider that are configured as part of the service package. This includes all production status-related notifications for that Functional Resource Type.

When the production status of the Return All Frames service associated with the return X-Band symbol stream transitions from ‘halted’ to ‘running’, the MD-CSTS provider invokes the NOTIFY operation of the Notification procedure instance to signal the occurrence of the event with the Event Name

[[{rtnAfTsProvider}: 1]: {rtnAfProdStatChange}].

This event carries the current value (‘running’) of the production status of the Return All Frames TS Provider associated with the return X-Band symbol stream.

When the production status of the Return All Frames service associated with the return S-Band symbol stream transitions from ‘halted’ to ‘running’, the MD-CSTS provider invokes the NOTIFY operation of the Notification procedure instance to signal the occurrence of the event with the Event Name

[[{`rtnAfTsProvider`}: 2]: {`rtnAfProdStatusChange`}].

This event carries the current value (‘running’) of the production status of the Return All Frames TS Provider associated with the return S-Band symbol stream.

2.5.3.8 Fourth Instance of Notification Procedure

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS starts the fourth Notification procedure instance to subscribe to the notifiable events of the following Functional Resource Type on occurrences of changes in the production status of the production processes for the Forward CLTU TS Provider:

{fwdCltuTsProvider}.

By using the Functional Resource Type in the subscription, the user subscribes to all event notifications for the Forward CLTU TS Provider instance that is configured as part of the service package. This includes all production status-related notifications for that Functional Resource Type.

When the production status of the Forward CLTU service associated with the forward S-Band symbol stream transitions from ‘configured’ to ‘operational’, the MD-CSTS provider invokes the NOTIFY operation of the Notification procedure instance to signal the occurrence of the event with the Event Name:

[[{`fwdCltuTsProvider`}: 1]: {`fwdCltuTsProdStatChange`}].

This event carries the current value (‘operational’) of the production status of the Forward CLTU TS Provider associated with the forward S-Band symbol stream.

2.5.3.9 Information Query Procedure

2.5.3.9.1 General

The Information Query procedure of the MD-CSTS can be used to retrieve the current values of parameters of Functional Resources that are configured as part of the service package. The Information Query procedure of the MD-CSTS can also be used to query the procedure configuration parameters of the Association Control, On-Change-Option Cyclic Report, Information Query, and Notification procedures of the MD-CSTS instance executing those procedures.

2.5.3.9.2 Query of Functional Resource Parameters

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS invokes the GET operation of the Information Query Procedure instance to request the current values of the `antPointingMode` parameter for both the S-Band and X-Band antennas. The GET invocation contains the Parameter Label [{`antenna`}: {`antPointingMode`}], which causes the MD service instance to return the parameter values for both configured antennas:

– [[{`antenna`}: 1]: {`antPointingMode`}] for the X-Band antenna; and

– [[{`antenna`}: 2]: {`antPointingMode`}] for the S-Band antenna.

NOTE – The GET operation of the Information Query procedure may be invoked at any time that the MD-CSTS instance is in the ‘bound’ state. In particular, the Information Query procedure may be used regardless of the state of the On-Change-Option Cyclic Report prime procedure instance. That is, the Information Query procedure may be used when the MD-CSTS instance is in either the ‘bound.ready’ or ‘bound.active’ state.

2.5.3.9.3 Query of Procedure Parameters

At some time in the period that the MD-CSTS is bound, the user of the MD-CSTS invokes the GET operation of the Information Query procedure to retrieve the contents of the Label List Set used by the On-Change-Option Cyclic Report procedure. The GET invocation contains the Parameter Name [[{`ocoCyclicReport`}: ‘prime procedure instance’]: {`pCRnamedLabelLists`}], which causes the MD service instance to return the Label List Set containing the single Label List that has been defined:

```
{   {   "defaultLabelList";
        TRUE;
        [{fwdCltuTsProvider}: {fwdCltuTsNumberOfCltusRadiated}];
        [{rtnAfTsProvider}: {rtnAfNumberOfFramesDelivered}]
    }
}
```

NOTES

1 The On-Change-Option Cyclic Report procedure inherits the `pCRnamedLabelLists` procedure configuration Parameter Identifier from the parent Framework Cyclic Report procedure. The OID for the Cyclic Report procedure configuration Parameter Identifier `pCRnamedLabelLists` is specified in F3.16 of reference [1].

2 There is only one set of Parameter Label Lists for all instances of the On-Change-Option Cyclic Report and Information Query procedures for all instances of those procedures in an MD-CSTS instance. Both the `pCRnamedLabelLists` parameter of the On-Change-Option Cyclic Report procedure and the `pCRnamedLabelLists` parameter of the Information Query procedure are populated from the single `mdNamedLabelLists` service management parameter. Thus the use of the On-Change-Option Cyclic Report prime procedure instance `pCRnamedLabelLists` parameter is not uniquely required to retrieve the Label List Set. The `pCRnamedLabelLists` parameter of any instance of the On-Change-Option Cyclic Report procedure could be used. Furthermore, the `pIQnamedLabelLists` parameter of the Information Query procedure could also be used to retrieve the same Label List Set.

3 The user of the MD service may also query Label List contents by procedure instance identifier. If the query is made on the procedure instance identifier, all procedure parameters for the specified instance of the specified procedure type are reported. In the case of this scenario, the GET invocation on the prime procedure instance of the On-Change-Option Cyclic Report procedure would return all procedure parameters of that procedure instance. The On-Change-Option Cyclic Report procedure has only one parameter (`pCRnamedLabelLists`), so the contents of `pCRnamedLabelLists` for the CR prime procedure instance would be reported. However, since the Label Lists are common to all instances of the procedure type, the same Label List Set would be returned regardless of which procedure instance is specified in the GET invocation.

4 The user of the MD service may also query Label List contents by procedure type. If the query is made on the procedure type, all procedure parameters for all configured instances of the specified procedure type are reported. In the case of this scenario, where four instances of On-Change-Option Cyclic Report procedure are configured (even though only three are activated), the GET invocation on the On-Change-Option Cyclic Report procedure would return four sets of procedure parameters, so the contents of `pCRnamedLabelLists` for each of the four CR procedure instances would be reported. However, since the Label Lists are common to all instances of the procedure type, four copies of the same Label List Set would be returned.

2.5.3.10 Stopping the MD-CSTS Procedures and Unbinding the MD-CSTS Instance

At some times prior to the end of the service instance provision period of the Monitored Data service instance, the user invokes (in no particular order) the STOP operations for three On-Change-Option Cyclic Report Procedure instances and the four instances of the Notification procedure.

NOTE – The MD-CSTS user may normally choose to stop the secondary On-Change-Option Cyclic Report and Notification procedure instances before stopping the prime procedure instance. However, this is not required.

After all On-Change-Option Cyclic Report Procedure instances and all Notification procedure instances have been stopped, but still prior to the end of the service instance provision period of the Monitored Data service instance, the user invokes the UNBIND operation of the Association Control procedure instance to transition the Monitored Data service instance to the ‘unbound’ state.

3 COMPOSITION OF THE MONITORED DATA CROSS SUPPORT TRANSFER SERVICE

3.1 OVERVIEW

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

The Object Identifiers for the Monitored Data service are specified in annex B.

3.2 PROCEDURES OF THE MONITORED DATA CROSS SUPPORT TRANSFER SERVICE

3.2.1 The Monitored Data transfer service shall be composed of the Association Control, On-Change-Option Cyclic Report, Notification, and Information Query procedures.

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

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

3.2.4 The version number of the Association Control procedure shall be the same as the version number of the Association Control procedure from reference [1].

3.2.5 The On-Change-Option Cyclic Report procedure shall be the prime procedure for the Monitored Data transfer service.

NOTE – Being the prime procedure implies that at least one instance of the procedure is mandatory.

3.2.6 The On-Change-Option Cyclic Report procedure shall be refined and extended from the Cyclic Report procedure defined in reference [1].

3.2.7 The version number of the On-Change-Option Cyclic Report procedure shall be ‘1’.

3.2.8 There shall be zero or more secondary instances of the On-Change-Option Cyclic Report procedure.

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

3.2.9 The Information Query procedure shall be refined from the Information Query procedure defined in reference [1].

3.2.10 The version number of the Information Query procedure shall be ‘1’.

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

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

3.2.12 The Notification procedure shall be refined from the Notification procedure defined in reference [1].

3.2.13 The version number of the Notification procedure shall be '1'.

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

NOTES

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

2 Table 3-1 summarizes the procedures that constitute the Monitored Data transfer service, where (a) the '[P]' in the *Procedure* row designates the On-Change-Option Cyclic Report as the prime procedure; (b) *Version* = '-' indicates that the version number of the service procedure is the same as that of the *Cross Support Transfer Service—Specification Framework* procedure for those procedures that are directly adopted, and *Version* = '1' indicates the version of the refined and extended service procedures (On-Change-Option Cyclic Report, Information Query, and Notification); (c) *No. of Instances* indicates the minimum and maximum number of allowed instances of each procedure type; (d) *Specification Approach* indicates which procedures are directly adopted or refined and extended; and (e) *Source* indicates the *Cross Support Transfer Service—Specification Framework* procedure from which the service procedure is adopted or refined and extended.

Table 3-1: Monitored Data Transfer Service Procedures

Procedure	Association Control	On-Change-Option Cyclic Report [P]	Information Query	Notification
Version	-	1	1	1
No. of Instances	1..1	1..*	0..1	0..*
Specification Approach	adopted	refined-and-extended	refined	refined
Source	Reference [1]: Association Control (4.3)	Reference [1]: Cyclic Report (4.10)	Reference [1]: Information Query (4.9)	Reference [1]: Notification (4.11)

3.3 MONITORED DATA CROSS SUPPORT TRANSFER SERVICE STATE MACHINE

The Monitored Data Cross Support Transfer Service state machine shall conform to the state machine for a CSTS with a stateful prime procedure instance, as defined in the CSTS Framework (G3 of reference [1]).

4 ON-CHANGE-OPTION CYCLIC REPORT PROCEDURE

4.1 DISCUSSION

4.1.1 PURPOSE

The On-Change-Option Cyclic Report procedure of the Monitored Data service is used to cyclically report monitored parameter values for all Functional Resources configured as part of a service package.

The On-Change-Option Cyclic Report procedure of the Monitored Data service may be used to cyclically report the procedure configuration parameters of the Association Control procedure, On-Change-Option Cyclic Report procedure, the Information Query procedure, and the Notification procedure of the MD-CSTS instance that executes those procedures. Support for cyclic reporting of procedure configuration parameters is optional.

Each instance of the On-Change-Option Cyclic Report procedure may optionally be set to report parameter values only if they have changed since the previous cyclic report.

4.1.2 CONCEPT

The concept of the On-Change-Option Cyclic Report procedure is the same as that of the parent *Cross Support Transfer Service—Specification Framework* Cyclic Report procedure, with three differences.

The first difference is that the On-Change-Option Cyclic Report adds the optional capability to report parameter values only when they have changed from the values that were previously (cyclically) reported for them.

NOTE – When the on-change option is invoked, if the value of a parameter changes between cyclic reports but changes back to the previously reported value at the time of the next cyclic report, no value is reported for that parameter. Only if the value of the parameter is different from the previously reported value *at the time that the next cyclic report is generated* will a changed value for that parameter be included in the cyclic report.

The second difference concerns what is reported when Parameter Labels or Functional Resource Types are used to subscribe to functional resource monitored parameters:

– When a Parameter Label is used to subscribe to functional resource monitored parameters, the On-Change-Option Cyclic Report procedure reports the values of all functional resource monitored parameters that have that Parameter Label for all instances of the associated Functional Resource Type that are configured as part of the service package. This includes Parameter Labels that are represented by named and default lists.

– When a Functional Resource Type is used to subscribe to functional resource monitored parameters, the On-Change-Option Cyclic Report procedure reports the values of all functional resource monitored parameters of all instances of that Functional Resource Type that are configured as part of the service package.

The third difference is that cyclic reporting of procedure configuration parameters is optional for implementations of the On-Change-Option Cyclic Report procedure. As described in reference [1], the Cyclic Report procedure has access to all configuration parameters of the procedures that constitute a CSTS that implements the Cyclic Report. In the case of the MD-CSTS, procedure configuration parameters of the procedures that constitute the MD service (Association Control, Cyclic Report, Information Query, and Notification) are all static for the duration of the MD service instance. Cyclic reporting of these parameters would result in the same information being reported every cycle. Implementers should consider the utility of repeatedly reporting the same values when deciding whether or not to implement the optional cyclic reporting of configuration parameters.

4.2 PROCEDURE TYPE IDENTIFIER

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

4.3 EXTENSION

The On-Change-Option Cyclic Report procedure extends the START operation of the *Cross Support Transfer Service—Specification Framework* Cyclic Report procedure by adding the `on-change-option` parameter.

The On-Change-Option Cyclic Report procedure refines the behavior of the *Cross Support Transfer Service—Specification Framework* Cyclic Report procedure by modifying the behavior of the procedure when Functional Resource Parameter Labels or Functional Resource Types are subscribed. The behavior is also refined to allow subscription to and reporting of procedure configuration parameters to be optional on an implementation basis.

4.4 BEHAVIOR

NOTES

1 In the following paragraphs, the phrase ‘the service instance that executes the Cyclic Report procedure’ is to be interpreted as referring to the Monitored Data service.

2 In the following paragraphs, the term ‘Cyclic Report procedure’ is to be interpreted as referring to the On-Change-Option Cyclic Report procedure.

4.4.1 The Stopping and Terminating behaviors of the On-Change-Option Cyclic Report procedure shall be the same as those of the Cyclic Report procedure as specified in reference [1].

4.4.2 The Starting behavior of the On-Change-Option Cyclic Report procedure shall be the same as that of the Cyclic Report procedure as specified in reference [1], except for the refinements in the following subparagraphs.

4.4.2.1 The specification in paragraph 4.10.3.1.3 c) shall be replaced with the following:

c) if the `list-of-parameters` parameter contains one Functional Resource Type that is configured as part of the service package;

4.4.2.2 The specification in paragraph 4.10.3.1.3 d) shall be replaced with the following:

d) if the `list-of-parameters` parameter contains the name of one Functional Resource instance that is configured as part of the service package;

4.4.2.3 The specification in paragraph 4.10.3.1.3 e) shall be replaced with:

e) if (1) the implementation of the service supports cyclic reporting of procedure configuration parameters and (2) the `list-of-parameters` parameter contains one procedure type that is associated with the service instance that executes the Cyclic Report procedure;

4.4.2.4 The specification in paragraph 4.10.3.1.3 f) shall be replaced with:

f) if (1) the implementation of the service supports cyclic reporting of procedure configuration parameters and (2) the `list-of-parameters` parameter contains the procedure instance identifier of a procedure instance that is associated with the service instance that executes the Cyclic Report procedure;

4.4.2.5 The specification in paragraph 4.10.3.1.3 g) shall be replaced with:

g) if (1) the `list-of-parameters` parameter contains one or more Functional Resource Parameter Names or Functional Resource Parameter Labels, and (2) every one of these names or labels is the name or label of a parameter of a Functional Resource instance that is configured as part of the service package;

4.4.2.6 The specification in paragraph 4.10.3.1.3 h) shall be replaced with:

h) if (1) the implementation of the service supports cyclic reporting of procedure configuration parameters, (2) the `list-of-parameters` parameter contains one or more procedure configuration Parameter Names or Parameter Labels, and (3) every one of the names or labels is the name or label of a parameter of a configured procedure instance that is associated with the service instance that executes the Cyclic Report procedure.

4.4.3 The Transferring Data behavior of the On-Change-Option Cyclic Report procedure shall be the same as that of the Cyclic Report procedure as specified in reference [1], except for the refinements specified in the following subparagraphs.

4.4.3.1 Add the following refined specifications to the specification in paragraph 4.10.3.2.3:

a) if the `on-change-option` parameter value is FALSE, the service provider shall send the complete set of qualified parameters designated by the `list-of-parameters` in every TRANSFER-DATA invocation;

b) if the `on-change-option` parameter value is TRUE:

   1) in the first TRANSFER-DATA invocation sent following the activation of the On-Change-Option Cyclic Report procedure instance, the service provider shall send the complete set of qualified parameters designated by the `list-of-parameters`;

   2) in every subsequent TRANSFER-DATA invocation, the service provider shall send qualified parameters designated by the `list-of-parameters` for only those parameters whose values and/or parameter qualifiers have changed since they were last reported in a TRANSFER-DATA invocation that was sent during the current active period of the On-Change-Option Cyclic Report procedure.

   NOTE – On-Change-Option reporting applies to changes in parameter qualifiers as well as to changes in valid values. For example, if the parameter qualifier of a parameter becomes unavailable, the qualified parameter with qualifier ‘unavailable’ will be sent in the next report, and no other qualified parameters will be sent for that parameter until the value of the parameter qualifier changes to something else.

4.4.3.2 The specification in paragraph 4.10.3.2.4 a) 1) shall be replaced with:

   1) for each Functional Resource Parameter Label in the default list, the service provider shall deliver the qualified parameter for that label for each instance of the given Functional Resource Type that is configured as part of the service package;

4.4.3.3 The specification in paragraph 4.10.3.2.4 b) 1) shall be replaced with the following:

   1) for each Functional Resource Parameter Label in the named list, the service provider shall deliver the qualified parameter for that label for each instance of the given Functional Resource Type that is configured as part of the service package;

4.4.3.4 The specification in paragraph 4.10.3.2.4 c) shall be replaced with:

c) contains a Functional Resource Type, the service provider shall deliver for each Parameter Label associated with that Functional Resource Type the qualified

parameter for that label of each Functional Resource instance of the given type that is configured as part of the service package;

4.4.3.5 The specification in paragraph 4.10.3.2.4 g) shall be replaced with:

g) contains any Parameter Labels for Functional Resource parameters, the service provider shall deliver the qualified parameter for that label for each instance of the given Functional Resource Type that is configured as part of the service package.

4.5 REQUIRED OPERATIONS

4.5.1 The On-Change-Option Cyclic Report procedure shall use the STOP and TRANSFER-DATA operations of the Cyclic Report procedure as specified in reference [1] without extension or refinement.

4.5.2 The On-Change-Option Cyclic Report procedure shall use the START operation of the Cyclic Report procedure as extended in 4.5.3.

NOTE – Table 4-1 summarizes the operations of the On-Change-Option Cyclic Report procedure of the Monitored Data service.

Table 4-1: On-Change-Option Cyclic Report Procedure Required Operations

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

4.5.3 START (CONFIRMED)

4.5.3.1 Invocation, Return, and Parameters

In addition to the parameters of the Cyclic Report START operation as defined in reference [1], the extension parameter specified in table 4-2 shall be present in the START invocation of the On-Change-Option Cyclic Report procedure.

Table 4-2: START Extension Parameter

Extension Parameter	Invocation	Return
on-change-option	M	

4.5.3.2 Extension Parameter Syntax

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

4.5.3.3 `on-change-option`

4.5.3.3.1 The `on-change-option` parameter shall have one of the two following values:

a) TRUE: For each subscribed parameter, a qualified parameter is to be sent only when the value or parameter qualifier is different from the previously sent qualified parameter.

b) FALSE: For each subscribed parameter, a qualified parameter is to be sent at every reporting cycle.

4.5.3.3.2 The `on-change-option` parameter shall be present in every START invocation.

4.6 CONFIGURATION PARAMETERS

The On-Change-Option Cyclic Report procedure adopts the configuration parameters of the Cyclic Report procedure as specified in reference [1] without addition or modification.

4.7 PROCEDURE STATE TABLE

The On-Change-Option Cyclic Report procedure adopts the state table of the Cyclic Report procedure as specified in reference [1] without addition or modification.

5 INFORMATION QUERY PROCEDURE

5.1 DISCUSSION

5.1.1 PURPOSE

The Information Query procedure of the Monitored Data service is used to query the current values of monitored parameters for any Functional Resource configured as part of a service package.

The Information Query procedure of the Monitored Data service is also used to query the procedure configuration parameters of the Association Control procedure, Cyclic Report procedure, the Information Query procedure, and the Notification procedure of the MD-CSTS instance that executes those procedures.

5.1.2 CONCEPT

The concept of the Information Query procedure is the same as that of the parent Information Query procedure as specified in reference [1], with one difference regarding what is reported when Functional Resource Parameter Labels or Functional Resource Types are used to request the values of monitored parameters.

– When a Parameter Label is used to request values of monitored parameters, the Information Query procedure returns the values of all monitored parameters that have that Parameter Label for all instances of the associated Functional Resource Type that are configured as part of the service package. This includes Parameter Labels that are represented by named and default lists.

– When a Functional Resource Type is used to request values of monitored parameters, the Information Query procedure returns the values of all monitored parameters of all instances of that Functional Resource Type that are configured as part of the service package.

5.2 PROCEDURE TYPE IDENTIFIER

The procedure type identifier for the MD-CSTS Information Query procedure shall be the same as the CSTS Framework Information Query procedure.

5.3 REFINEMENT

The Information Query procedure refines the behavior of the Information Query procedure as specified in reference [1] by specifying the behavior of the procedure when Parameter Labels or Functional Resource Types are used in queries.

5.4 BEHAVIOR

NOTE – The Getting Parameters behavior of the Information Query procedure as specified in reference [1] is the same as the behavior of the *Cross Support Transfer Service—Specification Framework* GET operation. The refinement of the behavior of the MD-CSTS Information Query procedure is specified as a refinement of the behavior of the *Cross Support Transfer Service—Specification Framework* GET operation.

5.4.1 The behavior of the Information Query GET operation shall be the same as that of the GET operation as specified in reference [1], except for the refinements specified in the following subparagraphs.

5.4.1.1 The specification in paragraph 3.12.1.3 c) of reference [1] shall be replaced with the following:

c) if the `list-of-parameters` parameter contains one Functional Resource Type that is configured as part of the service package;

5.4.1.2 The specification in paragraph 3.12.1.3 d) of reference [1] shall be replaced with the following:

d) if the `list-of-parameters` parameter contains one name of a Functional Resource instance that is configured as part of the service package;

5.4.1.3 The specification in paragraph 3.12.1.3 g) of reference [1] shall be replaced with the following:

g) if (1) the `list-of-parameters` parameter contains one or more Functional Resource Parameter Names or Functional Resource Parameter Labels, and (2) every one of these names or labels is the name or label of a parameter of a Functional Resource instance that is configured as part of the service package;

5.4.1.4 The specification in paragraph 3.12.1.4 a) 1) of reference [1] shall be replaced with:

1) for each Functional Resource Parameter Label represented by the default list, the service provider shall return the qualified parameter for that label for all instances of the given Functional Resource Type that are configured as part of the service package;

5.4.1.5 The specification in paragraph 3.12.1.4 b) 1) of reference [1] shall be replaced with the following:

1) for each Functional Resource Parameter Label in the named list, the service provider shall return the qualified parameter for that label for each Functional Resource instance of the given type that is configured as part of the service package;

5.4.1.6 The specification in paragraph 3.12.1.4 c) of reference [1] shall be replaced with:

c) contains a Functional Resource Type, then the service provider shall return the qualified parameters for all parameters of all Functional Resource instances of the given type that are configured as part of the service package;

5.4.1.7 The specification in paragraph 3.12.1.4 g) of reference [1] shall be replaced with:

g) contains any Functional Resource Parameter Labels, for each Functional Resource Parameter Label the service provider shall return the qualified parameter for that label for each of the Functional Resource instances of the given type that is configured as part of the service package;

5.4.2 The Terminating behaviors of the Information Query procedure shall be the same as that of the Information Query procedure as specified in 4.9.3.2 of reference [1].

5.5 REQUIRED OPERATIONS

The Information Query procedure shall use the GET operation of the Information Query procedure specified in reference [1] without extension or refinement.

NOTE – Table 5-1 summarizes the operations of the Information Query procedure of the Monitored Data service.

Table 5-1: Information Query Procedure Required Operations

Operations	Extended	Refined	Procedure Blocking/Non-Blocking
GET	N	N	Non-Blocking

5.6 CONFIGURATION PARAMETERS

The Information Query procedure adopts the configuration parameters of the Information Query procedure specified in reference [1] without addition or modification.

5.7 PROCEDURE STATE TABLE

The Information Query procedure adopts the state table of the Information Query procedure specified in reference [1] without addition or modification.

6 NOTIFICATION PROCEDURE

6.1 DISCUSSION

6.1.1 PURPOSE

The Notification procedure of the Monitored Data service is used to report event notifications for all Functional Resources configured as part of a service package.

NOTE – As described in reference [1], the Notification procedure has access to all procedure-specific notifiable events of the procedures that constitute a CSTS that implements the Notification procedure. In the case of the MD-CSTS, none of the procedures that are used by the service (Association Control, Cyclic Report, Information Query, and Notification) has any procedure-specific events that could be reported by the Notification procedure.

6.1.2 CONCEPT

The concept of the Notification procedure is the same as that of the parent Notification procedure specified in reference [1], with one difference regarding what is reported when Event Labels or Functional Resource Types are used to subscribe to functional resource notifiable events.

– When an Event Label is used to subscribe to notifiable events, the Notification procedure notifies the user of the occurrences of all events that have that Event Label for all instances of the associated Functional Resource Type that are configured as part of the service package. This includes Event Labels that are represented by named and default lists.

– When a Functional Resource Type is used to subscribe to notifiable events, the Notification procedure notifies the user of the occurrences of all events of all instances of that Functional Resource Type that are configured as part of the service package.

6.2 PROCEDURE TYPE IDENTIFIER

The procedure type identifier for the MD-CSTS Notification procedure shall be the same as for the CSTS Framework Notification procedure.

6.3 REFINEMENT

The Notification procedure refines the behavior of the Notification procedure specified in reference [1] by modifying the behavior of the procedure when Event Labels or Functional Resource Types are subscribed.

6.4 BEHAVIOR

6.4.1 The behavior of the Notification procedure shall be the same as that of the Notification procedure as specified in reference [1], except for the refinements specified in the following subparagraphs.

6.4.2 The specification in paragraph 4.11.3.1.3 c) shall be replaced with:

c) if the `list-of-events` parameter contains one Functional Resource Type that is configured as part of the service package;

6.4.3 The specification in paragraph 4.11.3.1.3 d) shall be replaced with:

d) if the `list-of-events` parameter contains one name of a Functional Resource instance that is configured as part of the service package;

6.4.4 The validation criterion in paragraph 4.11.3.1.3 e) does not apply to the Notification procedure of the Monitored Data service because none of the procedures that are used by the MD service have any procedure-specific events that could be reported by the Notification procedure.

6.4.5 The validation criterion in paragraph 4.11.3.1.3 f) does not apply to the Notification procedure of the Monitored Data service because none of the procedures that are used by the MD service have any procedure-specific events that could be reported by the Notification procedure.

6.4.6 The specification in paragraph 4.11.3.1.3 g) shall be replaced with:

g) if (1) the `list-of-events` parameter contains one or more Functional Resource Event Names or Functional Resource Event Labels, and (2) every one of those names or labels is the name or label of an event of a Functional Resource that is configured as part of the service package;

6.4.7 The validation criterion in paragraph 4.11.3.1.3 h) does not apply to the Notification procedure of the Monitored Data service because none of the procedures that are used by the MD service have any procedure-specific events that could be reported by the Notification procedure.

6.4.8 The specification in paragraph 4.11.3.2.2 a) 1) shall be replaced with:

1) for each Functional Resource Event Label in the default list, the service provider shall notify the occurrence of the event for that label for each Functional Resource instance of the given type that is configured as part of the service package;

6.4.9 The specification in paragraph 4.11.3.2.2 b) 1) shall be replaced with:

1) for each Functional Resource Event Label in the named list, the service provider shall notify the occurrence of the event for that label for each Functional

Resource instance of the given type that is configured as part of the service package;

6.4.10 The specification in paragraph 4.11.3.2.2 c) shall be replaced with:

c) contains a Functional Resource Type, then the service provider shall notify the occurrence of all events for all instances of that Functional Resource Type that are configured as part of the service package;

6.4.11 The specification in paragraph 4.11.3.2.2 g) shall be replaced with:

g) contains any Functional Resource Event Labels, then for each label the service provider shall notify the occurrence of the event for that label for each Functional Resource of the given type that is configured as part of the service package;

6.5 REQUIRED OPERATIONS

The Notification procedure shall use the START, STOP, and NOTIFY operations of the Notification procedure as specified in reference [1] without extension or refinement.

NOTE – Table 6-1 summarizes the operations of the Notification procedure.

Table 6-1: Notification Procedure Required Operations

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

6.6 CONFIGURATION PARAMETERS

The Notification procedure adopts the configuration parameters of the Notification procedure as specified in reference [1] without addition or modification.

6.7 PROCEDURE STATE TABLE

The Notification procedure adopts the state table of the Notification procedure as specified in reference [1] without addition or modification.

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

7.1 OVERVIEW

The BIND operation of the defines the `responder-port-identifier` parameter (3.4.2.2.4.3 in reference [1]) to be a service management parameter of each CSTS. Subsection 7.2, below, specifies the classifier to be used for the `responder-port-identifier` parameter for the MD-CSTS. The `parameterId` corresponding to this classifier is defined in the SANA Functional Resource Registry (reference [3]) subtree for the Monitored Data CSTS Provider Functional Resource.

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

The procedures specified in reference [1] define configuration parameters for the Framework procedures, but defer to the derived services the specification of the method by which each of those configuration parameters is to be set. Subsections 7.3 through 7.6 specify the method by which each of the Framework procedure configuration parameters is to be set for the MD-CSTS.

For each of the procedure configuration parameters that are specified to be a service management parameter, the classifier for each parameter is also specified. The `parameterId` corresponding to each such classifier is defined in the SANA Functional Resource Registry (reference [3]) subtree for the Monitored Data CSTS Provider Functional Resource.

7.2 `responder-port-identifier` SERVICE MANAGEMENT PARAMETER

The `responder-port-identifier` service management parameter (3.4.2.2.4.3 in reference [1]) shall have the classifier `mdResponderPortId`.

7.3 ASSOCIATION CONTROL PROCEDURE CONFIGURATION PARAMETERS

7.3.1 The `service-user-responding-timer` configuration parameter (4.3.5 in reference [1]) shall be a service management parameter with the classifier `mdServiceUserRespondingTimer`.

7.3.2 The `initiator-identifier` configuration parameter (4.3.5 in reference [1]) shall be a service management parameter with the classifier `mdInitiatorId`.

7.3.3 The `responder-identifier` configuration parameter (4.3.5 in reference [1]) shall be a service management parameter with the classifier `mdResponderId`.

7.3.4 The `service-instance-identifier` configuration parameter (4.3.5 in reference [1]) shall be a service management parameter with the classifier `mdServiceInstanceId`.

7.4 ON-CHANGE-OPTION CYCLIC REPORT PROCEDURE CONFIGURATION PARAMETERS

7.4.1 The set of named Parameter Label Lists that constitutes the `named-label-lists` configuration parameter (4.10.5 in reference [1]) shall be a service management parameter with the classifier `mdNamedLabelLists`.

7.4.2 The set of named Parameter Label Lists shall be available to all instances of On-Change-Option Cyclic Report and MD Information Query procedures of the MD-CSTS instance.

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

7.4.4 The default list of Parameter Labels shall be used as the default Parameter Label List by all instances of On-Change-Option Cyclic Report and Information Query procedures of the MD-CSTS instance.

7.4.5 The `minimum-allowed-delivery-cycle` configuration parameter (4.10.5 in reference [1]) shall be a service management parameter.

7.5 NOTIFICATION PROCEDURE CONFIGURATION PARAMETERS

NOTE – The Notification procedure is optional. If the MD Notification procedure is not available in an instance of the Monitored Data service, the values of the procedure configuration parameters associated with the MD Notification procedure are undefined.

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

7.5.2 The set of named Event Label Lists shall be available for use by the Notification procedure instance of the Monitored Data transfer service instance.

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

7.5.4 The default list of events shall be used as the default Event Label List by the Notification procedure instance of the Monitored Data transfer service instance.

7.6 INFORMATION QUERY PROCEDURE CONFIGURATION PARAMETERS

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

NOTE – Requirements on the composition of the set of named Parameter Label Lists with the classifier `mdNamedLabelLists` are specified in 7.4.2, 7.4.3, and 7.4.4.

8 MONITORED DATA SERVICE-SPECIFIC VERSIONS OF SERVICE-GENERIC PARAMETER AND EVENTS

8.1 OVERVIEW

Annex B of reference [1] specifies the following service-generic parameter and events for use by any CSTS:

a) A production status that can be monitored. The OID to be used for the parameter that contains the production status for every CSTS is specified in F4.17 of reference [1] with the classifier `svcProductionStatusVersion1`.

b) A production status change event that is to be emitted when the production status changes, as specified in 3.11.2.2.3.2 a) of reference [1]. The OID to be used for the production status change event for every CSTS is specified in F4.17 of reference [1] with the classifier `svcProductionStatusChangeVersion1`.

c) A production configuration change event that is to be emitted when any Functional Resource in the production experiences a configuration change, as specified in 3.11.2.2.3.2 b) of reference [1]. The OID to be used for the production configuration change event for every CSTS is specified in F4.17 of reference [1] with the classifier `svcProductionConfigurationChangeVersion1`.

Each CSTS is to provide its own label for the `production-status` parameter, production status change event, and production configuration change event.

The Monitored Data service supports the `production-status` parameter and the production status change event. However, the Monitored Data service does not support the production configuration change event because it is redundant with information that is already available through the MD service.

8.2 `mdSvcProductionStatus` PARAMETER

The `mdSvcProductionStatus` parameter shall contain the production status, with the Published Identifier `svcProductionStatusVersion1` as specified in the `CCSDS-CSTS-GENERIC-SERVICE-OBJECT-IDENTIFIERS` module specified in F4.17 of reference [1].

8.3 `mdSvcProductionStatusChange` EVENT

The `mdSvcProductionStatusEvent` event shall report production status changes, with the Published Identifier `svcProductionStatusChangeVersion1` as specified in the `CCSDS-CSTS-GENERIC-SERVICE-OBJECT-IDENTIFIERS` module specified in F4.17 of reference [1].

9 REFINEMENT OF DEFINITIONS OF FRAMEWORK PARAMETERS, EVENTS, DIRECTIVES, AND DIAGNOSTIC VALUES USED BY THE MONITORED DATA SERVICE

9.1 OVERVIEW

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

NOTE – With respect to the production status of the Monitored Data service, the definition of the 'production configuration change' event (see 3.11.2.2.3.2 b) in reference [1]) is not applicable because there are no configuration parameters of the Monitored Data Collection Functional Resource that can change dynamically.

9.2 `mdSvcProductionStatus` PARAMETER DEFINITION REFINEMENT

NOTE – This refined definition applies to the `mdSvcProductionStatus` parameter, which has the Published Identifier `svcProductionStatusVersion1` in the `CCSDS-CSTS-GENERIC-SERVICE-OBJECT-IDENTIFIERS` module specified in F4.17 of reference [1].

9.2.1 CONFIGURED

The definition of the 'configured' value of the `mdSvcProductionStatus` parameter shall be refined to mean that configuration of the resource performing the Monitored Data Collection function has been completed.

9.2.2 INTERRUPTED

The definition of the 'interrupted' value of the `mdSvcProductionStatus` parameter shall be refined to mean that the resource performing the Monitored Data Collection function has been stopped because of a condition that may be temporary.

9.2.3 HALTED

The definition of the 'halted' value of the `mdSvcProductionStatus` parameter shall be refined to mean that the resource performing the Monitored Data Collection function has been stopped by management action.

9.2.4 OPERATIONAL

The definition of the ‘operational’ value of the `mdSvcProductionStatus` parameter shall be refined to mean that the resource performing the Monitored Data Collection function has changed to ‘operational’.

9.3 `mdSvcProductionStatusChange` EVENT DEFINITION REFINEMENT

NOTE – This refined definition applies to the `mdSvcProductionStatusChange` event, which has the Published Identifier `svcProductionStatusChangeVersion1` in the `CCSDS-CSTS-GENERIC-SERVICE-OBJECT-IDENTIFIERS` module specified in F4.17 of reference [1].

The definition of the `event-value` of the `mdSvcProductionStatusChange` event shall be refined to mean that the change refers to the resource performing the Monitored Data Collection function.

9.4 DIAGNOSTIC VALUE DEFINITION REFINEMENT

9.4.1 The definition of the ‘unknown Functional Resource Type’ `diagnostic` value used by the On-Change-Option Cyclic Report procedure START operation (see 4.5.3, originally defined by 4.10.4.1.3.1 c) in reference [1]) and the Information Query procedure GET operation (see 5.5, originally defined by 3.12.2.4.1 c) in reference [1]) 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`.

9.4.2 The definition of the ‘unknown Functional Resource Name’ `diagnostic` value used by the On-Change-Option Cyclic Report procedure START operation (see 4.5.3, originally defined by 4.10.4.1.3.1 d) in reference [1]) and the Information Query procedure GET operation (see 5.5, originally defined by 3.12.2.4.1 d) in reference [1]) shall be refined as follows:

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

9.4.3 The definition of the ‘unknown parameter identifier’ `diagnostic` used by the On-Change-Option Cyclic Report procedure START operation (see 4.5.3, originally defined by 4.10.4.1.3.1 g) in reference [1]) and the Information Query procedure GET operation (see 5.5, originally defined by 3.12.2.4.1 g) in reference [1]) shall be refined as follows:

‘unknown parameter identifier’ – one or more Parameter Identifiers contained in the `list-of-parameters` parameter are unknown to the service provider for one of the following reasons:

1) the Functional Resource specified as part of the Parameter Name is not associated with any Functional Resource that is configured as part of the service package;

2) the Functional Resource Type as part of the Parameter Label is not associated with any Functional Resource that is configured as part of the service package;

3) a parameter with the given Published Identifier does not exist for the specified Functional Resource instance or Type.

The list of unknown Parameter Names or Parameter Labels shall be returned with the `diagnostic`. For each unknown Parameter Identifier that is contained in a Parameter Name in the `list-of-parameters` parameter, the Parameter Name shall be returned. For each unknown Parameter Identifier that is contained in a Parameter Label in the `list-of-parameters` parameter, the Parameter Label shall be returned.

9.4.4 The definition of the ‘unknown Functional Resource Type’ `diagnostic` used by the Notification procedure START operation (see 6.5, originally defined by 4.11.4.1.3 c) in reference [1]) shall be refined as follows:

‘unknown Functional Resource Type’—the Functional Resource Type contained in the `list-of-events` 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.

9.4.5 The definition of the ‘unknown Functional Resource Name’ `diagnostic` used by the Notification procedure START operation (see 6.5, originally defined by 4.11.4.1.3 d) in reference [1]) shall be refined as follows:

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

9.4.6 The definition of the ‘unknown event identifier’ diagnostic used by the Notification procedure START operation (see 6.5, originally defined by 4.11.4.1.3 g) in reference [1]) shall be refined as follows:

‘unknown event identifier’—one or more Event Identifiers contained in the list-of-events parameter are unknown to the service provider for one of the following reasons:

1) the Functional Resource specified as part of the Event Name is not associated with any Functional Resource that is configured as part of the service package;

2) the Functional Resource Type specified as part of the Event Label is not associated with any Functional Resource that is configured as part of the service package;

3) an event with the given Published Identifier does not exist for the specified Functional Resource instance or type.

9.4.7 The list of unknown Event Names or Event Labels shall be returned with the `diagnostic`. For each unknown Event Identifier that is contained in an Event Name in the `list-of-events` parameter, the Event Name shall be returned. For each unknown Event Identifier that is contained in an Event Label in the `list-of-events` parameter, the Event Label shall be returned.

ANNEX A

IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA

(NORMATIVE)

A1 INTRODUCTION

A1.1 OVERVIEW

This annex provides the Implementation Conformance Statement (ICS) Requirements List (RL) for an implementation of the *Cross Support Transfer Services—Monitored Data Service*, CCSDS 922.1-B-1, April 2017. CCSDS 922.1-B-1 specifies the requirements on the provider of the Monitored Data Cross Support Transfer Service.

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

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

a) the implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight;

b) a supplier or potential acquirer of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;

c) a user or potential user of the implementation, as a basis for initially checking the possibility of interworking with another implementation (it should be noted that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSes);

d) a tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A1.2 ABBREVIATIONS AND CONVENTIONS

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

Item Column

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

Feature Column

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

Status Column

The status column uses the following notations:

a) M mandatory;

b) O optional;

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

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

e) X prohibited;

f) N/A not applicable.

Support Column Symbols

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

a) Y Yes, supported by the implementation;

b) N No, not supported by the implementation;

c) N/A Not applicable.

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

Allowed Values Column

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

Allowed Values Column Symbols

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

Supported Values Column

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

a) Y — Yes, the range/set defined in the Recommended Specification is fully supported by the implementation;
b) SV<n> — The range/set defined in the Recommended Specification is not fully supported by the implementation. The supported subset is documented below the table.

A1.3 INSTRUCTIONS FOR COMPLETING THE RL

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

A2 PICS PROFORMA FOR THE MONITORED DATA CSTS PROTOCOL (CCSDS 922.1-B-1)

A2.1 GENERAL INFORMATION

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

Table A-1: Identification of PICS

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

Table A-2: Identification of Implementation under Test

Implementation name	
Implementation version	
Special Configuration	
Other Information	

Table A-3: Identification of Supplier

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

Table A-4: Identification of Specification

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

A2.2 REQUIREMENTS LIST

This subsection provides the Requirement Lists for the elements specified in this Recommended Standard.

Table A-5: Required Procedures

Procedures				
Item	**Description**	**Reference**	**Status**	**Support**
proc-1	Association Control	subsection 4.3 of reference [1]	M	
proc-2	Unbuffered Data Delivery	subsection 4.4 of reference [1]	M	
proc-3	Information Query	subsection 4.9 of reference [1], section 5	O	
proc-4	Cyclic Report	subsection 4.10 of reference [1]	M	
proc-5	Notification	subsection 4.11 of reference [1], section 6	O	
proc-6	On-Change-Option Cyclic Report	section 4	M	

The Unbuffered Data Delivery and Cyclic Report procedures are mandatory in the sense that the On-Change-Option Cyclic Report procedure (which is mandatory) is derived from the Cyclic Report procedure, which is in turn derived from the Unbuffered Data Delivery procedure. In this MD-CSTS ICS, all requirements for the Unbuffered Data Delivery procedure and Cyclic Report procedure are covered by the requirements for the On-Change-Option Cyclic Report procedure.

Table A-6: Required PDUs

Item	PDU	Reference	Service-Provider-System		Service-User-System	
			Status	Support	Status	Support
pdu-1	BindInvocation	subsection F4.5 of reference [1]	M		M	
pdu-2	BindReturn	subsection F4.5 of reference [1]	M		M	
pdu-3	PeerAbortInvocation	subsection F4.5 of reference [1]	M		M	
pdu-4	UnbindInvocation	subsection F4.5 of reference [1]	M		M	
pdu-5	UnbindReturn	subsection F4.5 of reference [1]	M		M	
pdu-6	GetInvocation	subsection F4.4 of reference [1]	C1		C1	
pdu-7	GetReturn	subsection F4.4 of reference [1]	C1		C1	
pdu-8	NotifyInvocation	subsection F4.4 of reference [1]	C2		C2	
pdu-9	StartInvocation	subsection F4.4 of reference [1]	M		M	
pdu-10	StartReturn	subsection F4.4 of reference [1]	M		M	
pdu-11	StopInvocation	subsection F4.4 of reference [1]	M		M	
pdu-12	StopReturn	subsection F4.4 of reference [1]	M		M	
pdu-13	TransferDataInvocation	subsection F4.4 of reference [1]	M		M	

C1 IF proc-3 THEN M ELSE N/A

C2 IF proc-5 THEN M ELSE N/A

Table A-7: BIND Invocation Parameters

Parameters of the BindInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
bindInv-1	invokerCredentials	subsection F4.3 of reference [1]	M			
bindInv-2	invokeId	subsection F4.3 of reference [1]	M			
bindInv-3	procedureInstanceId	subsection F4.3 of reference [1]	M		AV1	
bindInv-4	initiatorIdentifier	subsection F4.5 of reference [1]	M			
bindInv-5	responder-PortIdentifier	subsection F4.5 of reference [1]	M			
bindInv-6	serviceType	subsection F4.5 of reference [1]	M			
bindInv-7	versionNumber	subsection F4.5 of reference [1]	M			
bindInv-8	serviceInstance-Identifier	subsection F4.5 of reference [1]	M			
bindInv-9	bindInvocation-Extension	subsection F4.5 of reference [1]	M		not used	

AV1 The procedureType element of parameter bindInv-3 must be set to {associationControl} (see F4.1 in reference [1]). The procedureRole element of parameter bindInv-3 must be set to ‘associationControl’.

The parameters in table A-7 that reference F4.5 of reference [1] are contained in the BindInvocation type defined in F4.5 of reference [1].

The parameters bindInv-1, bindInv-2, and bindInv-3 are contained in the complex standardInvocationHeader parameter in the BindInvocation type defined in F4.5 of reference [1]. This parameter is of the type StandardInvocationHeader that is defined in F4.3 of reference [1].

Table A-8: BIND Return Parameters

Parameters of the BindReturn PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
bindRet-1	performerCredentials	subsection F4.3 of reference [1]	M			
bindRet-2	invokeId	subsection F4.3 of reference [1]	M			
bindRet-3	result	subsection F4.3 of reference [1]	M			
bindRet-4	positive	subsection F4.3 of reference [1]	C3		not used	
bindRet-5	diagnostic	subsection F4.3 of reference [1]	C4		AV2	
bindRet-6	negExtension	subsection F4.3 of reference [1]	C4		not used	
bindRet-7	responderIdentifier	subsection F4.5 of reference [1]	M			

C3 IF bindRet-3 = ‘positive’ THEN M ELSE X

C4 IF bindRet-3 = ‘negative’ THEN M ELSE X

AV2 For the negative BIND return the `diagnostic` parameter is extended by the type AssocBindDiagnosticExt defined in F4.5 of reference [1]. Therefore the parameter bindRet-5 may have (a) any value defined for the `Diagnostic` type in F4.3 of reference [1] except diagnosticExtension or (b) any value defined by ‘diagnosticExtension’: ‘acBindDiagExt’: ‘AssocBindDiagnosticExt’ defined in F4.5 of reference [1] except ‘assocBindDiagnosticExtExtension’.

All parameters of the BIND return PDU except bindRet-7 are contained in the complex type StandardReturnHeader that is defined in F4.3 of reference [1].

Table A-9: PEER-ABORT Invocation Parameters

Parameters of the PeerAbortInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
peerAbortInv-1	diagnostic	subsection F4.5 of reference [1]	M		40..51, 126	

Table A-10: UNBIND Invocation Parameters

Parameters of the UnbindInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
unbindInv-1	invokerCredentials	subsection F4.3 of reference [1]	M			
unbindInv-2	invokeId	subsection F4.3 of reference [1]	M			
unbindInv-3	procedureInstanceId	subsection F4.3 of reference [1]	M		AV3	
unbindInv-4	unbindInvocation-Extension	subsection F4.5 of reference [1]	M		not used	

AV3 The procedureType element of parameter unbindInv-3 must be set to {associationControl} (see F4.1 in reference [1]). The procedureRole element of parameter unbindInv-3 must be set to ‘associationControl’.

The parameters unbindInv-1, unbindInv-2, and unbindInv-3 are contained in the complex standardInvocationHeader parameter in the UnbindInvocation type defined in F4.5 of reference [1]. This parameter is of the type StandardInvocationHeader that is defined in F4.3 of reference [1].

Table A-11: UNBIND Return Parameters

Parameters of the UnbindReturn PDU						
Item	Parameter	Reference	Status	Support	Values	
					Allowed	Supported
unbindRet-1	performerCredentials	subsection F4.3 of reference [1]	M			
unbindRet-2	invokeId	subsection F4.3 of reference [1]	M			
unbindRet-3	result	subsection F4.3 of reference [1]	M		AV4	

AV4 The value of the parameter unbindRet-3 for the UNBIND return PDU shall always be set to the value ‘positive’: ‘notUsed’; i.e., the result is always positive and not extended.

Table A-12: GET Invocation Parameters

Parameters of the GetInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
getInv-1	invokerCredentials	subsection F4.3 of reference [1]	M			
getInv-2	invokeId	subsection F4.3 of reference [1]	M			
getInv-3	procedureInstanceId	subsection F4.3 of reference [1]	M		AV5	
getInv-4	listOfParameters	subsection F4.4 of reference [1]	M			
getInv-5	getInvocation-Extension	subsection F4.4 of reference [1]	M		not used	

AV5 The procedureType element of parameter getInv-3 must be set to {informationQuery} (see F4.1 in reference [1]). The value of the procedureRole element of the parameter getInv-3 is constrained to the value ‘secondary procedure’.

The parameters getInv-1, getInv-2, and getInv-3 are contained in the complex standardInvocationHeader parameter in the GetInvocation type defined in F4.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is defined in F4.3 of reference [1].

Table A-13: GET Return Parameters

Parameters of the GetReturn PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
getRet-1	performerCredentials	subsection F4.3 of reference [1]	M			
getRet-2	invokeId	subsection F4.3 of reference [1]	M			
getRet-3	result	subsection F4.3 of reference [1]	M			
getRet-4	positive	subsection F4.3 of reference [1]	C5			
getRet-5	qualifiedParameters	subsection F4.4 of reference [1]	C5		AV6	
getRet-6	getPosReturnExt-Extension	subsection F4.4 of reference [1]	C5		not used	
getRet-7	diagnostic	subsection F4.3 of reference [1]	C6		AV7	
getRet-8	negExtension	subsection F4.3 of reference [1]	C6		not used	

C5 IF getRet-3 = ‘positive’ THEN M ELSE N/A

C6 IF getRet-3 = ‘negative’ THEN M ELSE N/A

AV6 For the positive GET return the parameter getRet-5 is specified by ‘qualifiedParameters’: ‘QualifiedParametersSequence’. The type QualifiedParametersSequence is defined in F4.4 of reference [1].

AV7 For the negative GET return the parameter getRet-7 is extended by the type GetDiagnosticExt defined in F4.4 of reference [1]. Therefore the parameter getRet-7 may have (a) any standard value defined for the `Diagnostic` type in F4.3 of reference [1] except ‘diagnosticExtension’ or (b) any value defined by the extension ‘diagnosticExtension’: ‘getDiagnosticExt’: ‘GetDiagnosticExt’ defined in F4.4 of reference [1] except ‘getDiagnosticExtExtension’.

All parameters of the GET return PDU are contained in the complex parameter of the type StandardReturnHeader that is specified in F4.3 of reference [1]. Specific extensions are, however, specified in F4.4 of reference [1].

Table A-14: START Invocation Parameters

Parameters of the StartInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
startInv-1	invokerCredentials	subsection F4.3 of reference [1]	M			
startInv-2	invokeId	subsection F4.3 of reference [1]	M			
startInv-3	procedureInstanceId	subsection F4.3 of reference [1]	M		AV8	
startInv-4	startInvocation-Extension	subsection F4.4 of reference [1]	M		AV9	
startInv-5	deliveryCycle	subsection F4.12 of reference [1]	C7			
startInv-6	listOfParameters	subsection F4.12 of reference [1]	C7			
startInv-7	cyclicReportStart-InvocExtExtension	subsection F4.12 of reference [1]	C7		AV10	
startInv-8	onChangeOnly	annex B	C7			
startInv-9	onChangeOptData-CyclicReportStart-InvocExtExtension	annex B	C7		not used	
startInv-10	listOfEvents	subsection F4.13 of reference [1]	C8			
startInv-11	notificationStartInvoc-ExtExtension	subsection F4.13 of reference [1]	C8		not used	

C7 IF getRet-3 = ‘positive’ THEN M ELSE N/A

C8 IF getRet-3 = ‘negative’ THEN M ELSE N/A

AV8 The procedureType element of parameter startInv-3 must be set to either {ocoCyclicReport} (see annex B) or {notification} (see F4.1 in reference [1]).

If the procedureType element of parameter startInv-3 is set to {ocoCyclicReport}, the value of the procedureRole element of the parameter startInv-3 is constrained to one of the two values ‘prime procedure’ or ‘secondary procedure’.

If the procedureType element of parameter startInv-3 is set to {notification}, the

value of the procedureRole element of the parameter startInv-3 is constrained to the value ‘secondary procedure’.

AV9 If the procedureType element of the parameter startInv-3 has the value {ocoCyclicReport}, then the parameter startInv-4 shall be set to the value ‘crStartInvocExt’: ‘CyclicReportStartInvocExt’.

If the procedureType element of the parameter startInv-3 has the value {notification}, then the parameter startInv-4 shall be set to the value ‘nStartInvocExt’: ‘NotificationStartInvocExt’.

AV10 If the procedureType element of the parameter startInv-3 has the value {ocoCyclicReport}, then the parameter startInv-7 shall be set to the value ‘onChangeOptCyclicReportStartInvocExt’: ‘OnChangeOptCyclicReportStartInvocExt’.

The parameters startInv-1, startInv-2, and startInv-3 are contained in the complex standardInvocationHeader parameter in the StartInvocation type defined in F4.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is defined in F4.3 of reference [1].

Table A-15: START Return Parameters

Parameters of the StartReturn PDU						
Item	Parameter	Reference	Status	Support	Values	
					Allowed	Supported
startRet-1	performerCredentials	subsection F4.3 of reference [1]	M			
startRet-2	invokeId	subsection F4.3 of reference [1]	M			
startRet-3	result	subsection F4.3 of reference [1]	M			
startRet-4	positive	subsection F4.3 of reference [1]	C9		not used	
startRet-5	diagnostic	subsection F4.3 of reference [1]	C10		AV11	
startRet-6	negExtension	subsection F4.3 of reference [1]	C10		not used	

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

C10 IF startRet-3 = ‘negative’ THEN M ELSE N/A

AV11 If the procedureType element of the parameter startInv-3 of the associated START invocation has the value {ocoCyclicReport}, then the `diagnostic` parameter is extended by the type CyclicReportStartDiagnosticExt defined in F4.12 of reference [1]. Therefore the parameter startRet-5 may have in this case (a) any standard value defined for the `Diagnostic` type in F4.3 of reference [1] except ‘diagnosticExtension’; (b) any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’: ‘StartDiagnosticExt’ in F4.4 of reference [1] except ‘startDiagnosticExtExtension’; or (c) any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’: ‘StartDiagnosticExt’: ‘startDiagnosticExtExtension’: crStartDiagExt’: ‘CyclicReportStartDiagnosticExt’ defined in F4.12 of reference [1] except ‘cyclicReportStartDiagnosticExtExtension’.

If the procedureType element of the parameter startInv-3 of the associated START invocation has the value {notification}, then the `diagnostic` parameter is extended by the type NotificationStartDiagnosticExt defined in F4.13 of reference [1]. Therefore the parameter startRet-5 may have in this case (a) any standard value defined for the `Diagnostic` type in F4.3 of reference [1] except ‘diagnosticExtension’; (b) any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’:

‘StartDiagnosticExt’ in F4.4 of reference [1] except ‘startDiagnosticExtExtension’; or (c) any value defined by the extension ‘diagnosticExtension’: ‘startDiagnosticExt’: ‘StartDiagnosticExt’: ‘startDiagnosticExtExtension’: ‘nStartDiagExt’: ‘NotificationStartDiagnosticExt’ defined in F4.13 of reference [1] except ‘notificationStartDiagnosticExtExtension’.

All parameters of the START return PDU are contained in the complex parameter of the type StandardReturnHeader that is specified in F4.3 of reference [1]. Specific extensions are, however, specified in F4.4, F4.12, and F4.13 of reference [1], as appropriate.

Table A-16: STOP Invocation Parameters

Parameters of the StopInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
stopInv-1	invokerCredentials	subsection F4.3 of reference [1]	M			
stopInv-2	invokeId	subsection F4.3 of reference [1]	M			
stopInv-3	procedureInstanceId	subsection F4.3 of reference [1]	M		AV12	
stopInv-4	StopInvocation-Extension	subsection F4.4 of reference [1]	M		not used	

AV12 The procedureType element of parameter stopInv-3 must be set to either {ocoCyclicReport} (see annex B) or {notification} (see F4.1 of reference [1]).

If the procedureType element of parameter stopInv-3 is set to {ocoCyclicReport}, the value of the procedureRole element of the parameter stopInv-3 is constrained to one of the two values ‘prime procedure’ or ‘secondary procedure’.

If the procedureType element of parameter stopInv-3 is set to {notification}, the value of the procedureRole element of the parameter stopInv-3 is constrained to the value ‘secondary procedure’.

The parameters stopInv-1, stopInv-2, and stopInv-3 are contained in the complex standardInvocationHeader parameter in the StopInvocation type defined in F4.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is defined in F4.3 of reference [1].

Table A-17: STOP Return Parameters

Parameters of the StopReturn PDU						
Item	Parameter	Reference	Status	Support	Values	
					Allowed	Supported
stopRet-1	performerCredentials	subsection F4.3 of reference [1]	M			
stopRet-2	invokeId	subsection F4.3 of reference [1]	M			
stopRet-3	result	subsection F4.3 of reference [1]	M			
stopRet-4	positive	subsection F4.3 of reference [1]	C11		not used	
stopRet-5	diagnostic	subsection F4.3 of reference [1]	C12		AV13	
stopRet-6	negExtension	subsection F4.3 of reference [1]	C12		not used	

C11 IF stopRet-3 = ‘positive’ THEN M ELSE X

C12 IF stopRet-3 = ‘negative’ THEN M ELSE N/A

AV13 The parameter stopRet-5 may have any standard value defined for the `Diagnostic` type in F4.3 of reference [1] except ‘diagnosticExtension’.

Table A-18: NOTIFY Invocation Parameters

Parameters of the NotifyInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
notifyInv-1	invokerCredentials	subsection F4.3 of reference [1]	M			
notifyInv-2	invokeId	subsection F4.3 of reference [1]	M			
notifyInv-3	procedureInstanceId	subsection F4.3 of reference [1]	M		AV14	
notifyInv-4	eventTime	subsection F4.4 of reference [1]	M			
notifyInv-5	eventName	subsection F4.4 of reference [1]	M			
notifyInv-6	eventValue	subsection F4.4 of reference [1]	M		AV15	
notifyInv-7	notifyInvocation-Extension	subsection F4.4 of reference [1]	M		not used	

AV14 The procedureType element of parameter notifyInv-3 must be set to {notification} (see F4.1 of reference [1]). The value of the procedureRole element of the parameter notifyInv-3 must be set to the value ‘secondary procedure’.

AV15 The value of the notifyInv-6 parameter can be any value that can be expressed using the type SequenceOfQualifiedValues defined in F4.3 of reference [1] or ‘empty’, but it must not be set to ‘eventValueExtension’.

The parameters notifyInv-1, notifyInv-2, and notifyInv-3 are contained in the complex parameter standardInvocationHeader in the NotifyInvocation type defined in F4.4 of reference [1]. This parameter is of the type StandardInvocationHeader that is specified in F4.3 of reference [1].

Table A-19: TRANSFER-DATA Invocation Parameters

Parameters of the TransferDataInvocation PDU						
					Values	
Item	Parameter	Reference	Status	Support	Allowed	Supported
transferDataInv-1	invokerCredentials	subsection F4.3 of reference [1]	M			
transferDataInv-2	invokeId	subsection F4.3 of reference [1]	M			
transferDataInv-3	procedureInstanceId	subsection F4.3 of reference [1]	M		AV16	
transferDataInv-4	generationTime	subsection F4.4 of reference [1]	M			
transferDataInv-5	sequenceCounter	subsection F4.4 of reference [1]	M			
transferDataInv-6	data	subsection F4.4 of reference [1]	M		AV17	
transferDataInv-7	qualifiedParameters	subsection F4.12 of reference [1]	M			
transferDataInv-8	cyclicReportTransfer-DataInvocDataRef-Extension	subsection F4.12 of reference [1]	M		not used	
transferDataInv-9	transferData-InvocationExtension	subsection F4.4 of reference [1]	M		not used	

AV16 The procedureType element of parameter transferDataInv-3 must be set to {ocoCyclicReport} (see annex B). The value of the procedureRole element of the parameter transferDataInv-3 is constrained to one of the two values ‘prime procedure’ or ‘secondary procedure’.

AV17 The parameter transferDataInv-6 shall be set to the value ‘extendedData’: ‘crTransferDataInvocDataRef’: ‘CyclicReportTransferDataInvocDataRef’. The type CyclicReportTransferDataInvocDataRef is defined in F4.12 of reference [1].

ANNEX B

SERVICE OBJECT IDENTIFIERS MODULE

(NORMATIVE)

```
CCSDS-MONITORED-DATA-OBJECT-IDENTIFIERS
{  iso(1) identified-organization(3) standards-producing-organization(112)
   ccsds(4) css(4) csts(1) services(2) monitoredData(1)
   monitoredDataServiceModules(4) object-identifiers(1)
}

DEFINITIONS
IMPLICIT TAGS
::= BEGIN

EXPORTS  monitoredDataDerivedServices
,        monitoredDataExtendedServiceParameters
,        monitoredDataServiceProcedures
,        mdCstsProvider
,        mdCstsProviderParametersId
,        mdCstsProviderEventsId
;

IMPORTS  services
,        crossSupportFunctionalities
         FROM CCSDS-CSTS-OBJECT-IDENTIFIERS
;

-- ********************************************************

-- Root Object Identifiers of the Service

monitoredData  OBJECT IDENTIFIER ::=   {services 1}
monitoredDataDerivedServices  OBJECT IDENTIFIER ::= {monitoredData 1}
monitoredDataExtendedServiceParameters OBJECT IDENTIFIER    ::=
                                       {monitoredData 2}
monitoredDataServiceProcedures   OBJECT IDENTIFIER    ::=
                                       {monitoredData 3}
monitoredDataServiceModules   OBJECT IDENTIFIER ::= {monitoredData 4}

-- ********************************************************
-- Procedure Type Identifier
ocoCyclicReport   OBJECT IDENTIFIER ::= {monitoredDataServiceProcedures 1}

-- ********************************************************
-- Root Object Identifiers of the MD-CSTS Provider
-- Functional Resource Type
mdCstsProvider    OBJECT IDENTIFIER ::= {crossSupportFunctionalities 17}
mdCstsProviderParametersId OBJECT IDENTIFIER    ::= {mdCstsProvider 1}
mdCstsProviderEventsId OBJECT IDENTIFIER        ::= {mdCstsProvider 2}
mdCstsProviderDirectivesId OBJECT IDENTIFIER    ::= {mdCstsProvider 3}

END
```

ANNEX C

PROCEDURE – ON-CHANGE-OPTION CYCLIC REPORT PDUS

(NORMATIVE)

C1 ON-CHANGE-OPTION CYCLIC REPORT PDU MODULE

```
CCSDS-ON-CHANGE-OPTION-CYCLIC-REPORT-PDUS
{iso(1) identified-organization(3) standards-producing-organization(112)
ccsds(4) css(4) csts(1) services(2) monitoredData(1)
monitoredDataServiceModules(4) extensions(2) ocoCyclicReportPdus(1)
}

DEFINITIONS
IMPLICIT TAGS
::=   BEGIN

IMPORTS  Extended
         FROM CCSDS-CSTS-COMMON-TYPES

-- CCSDS-CSTS-COMMON-TYPES is defined in the
-- CSTS Specification Framework (reference [1])

         UnbufferedDataDeliveryPdu
         FROM CCSDS-CSTS-UNBUFFERED-DATA-DELIVERY-PDUS

-- CCSDS-CSTS-UNBUFFERED-DATA-DELIVERY-PDUS is defined in the
-- CSTS Specification Framework (reference [1])

         monitoredDataExtendedServiceParameters
         FROM CCSDS-MONITORED-DATA-OBJECT-IDENTIFIERS
;

-- The On-Change-Option Cyclic Report procedure is derived from the CSTS
-- SFW Cyclic Report procedure, which in turn is derived from the
-- Unbuffered Data Delivery procedure. It reuses the PDU defined in the
-- Unbuffered Data Delivery procedure:
-- UnbufferedDataDeliveryPdu type defined in the
-- CCSDS-CSTS-UNBUFFERED-DATA-DELIVERY-PDUS module of the CSTS
-- Specification Framework (reference [1]).

-- *****
-- START invocation extension parameters
-- START invocation is extended with the on-change-option parameter. The
-- Cyclic Report procedure specified in reference [1] extends the
-- parameters of the Unbuffered Data Delivery procdure START invocation.
-- Therefore this extension applies to the
-- CyclicReportStartInvocExt: cyclicReportStartInvocExtExtension
-- extension parameter that is defined in the CCSDS-CSTS-CYCLIC-REPORT-PDUS
-- module of the CSTS Specification Framework (reference [1]).
```

```
OnChangeOptCyclicReportStartInvocExt   ::=   SEQUENCE
{  onChangeOnly                                        BOOLEAN
,  onChangeOptCyclicReportStartInvocExtExtension      Extended
}

onChangeOptCyclicReportStartInvocExt   OBJECT IDENTIFIER  ::=
                                {monitoredDataExtendedServiceParameters 1}

-- START positive return parameters
-- No extension parameters are added to the START positive return. The CSTS
-- SFW Cyclic Report procedure does not extend the parameters of the START
-- positive return. Therefore
-- StartReturn(StandardReturnHeader): result: positive
-- (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 CSTS
-- SFW Cyclic Report procedure does not extend the parameters of the START
-- negative return. Therefore
-- StartReturn(StandardReturnHeader): result: negative: negExtension
-- (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
-- Cyclic Report procedure specified in reference [1] does not extend the
-- diagnostic values of the START negative return. Therefore the diagnostic
-- extension parameter CyclicReportStartDiagnosticsExt:
-- cyclicReportStartDiagnosticsExtExtension
-- (see CCSDS-CSTS-BUFFERED-DATA-DELIVERY-PDUS module in reference [1]) is
-- set to ‘notUsed’.

-- *****
-- STOP Invocation extension parameters
-- No extension parameters are added to the STOP invocation. The Cyclic
-- Report procedure specified in reference [1] does not extend the
-- parameters of the STOP invocation. Therefore
-- StopInvocation: stopInvocationExtension (see CCSDS-CSTS-COMMON-
-- OPERATIONS-PDUS module in reference [1]) is set to ‘notUsed’.

-- STOP positive return extension parameters
-- No extension parameters are added to the STOP positive return. The CSTS
-- SFW Cyclic Report procedure does not extend the parameters of the STOP
-- positive return. Therefore
-- StopReturn(StandardReturnHeader): result: positive
-- (see CCSDS-CSTS-COMMON-OPERATIONS-PDUS module in reference [1]) is set
-- to ‘notUsed’.

-- STOP negative return extension parameters
-- No extension parameters are added to the STOP negative return. The CSTS
-- SFW Cyclic Report procedure does not extend the parameters of the STOP
-- negative return. Therefore
-- StopReturn(StandardReturnHeader): result: negative: negExtension
-- (see CCSDS-CSTS-COMMON-OPERATIONS-PDUS module in reference [1]) is set
-- to ‘notUsed’.
```

```
-- STOP negative return extension diagnostics
-- No extension diagnostics added to the STOP negative return. The Cyclic
-- Report procedure specified in reference [1] does not extend the
-- diagnostic values of the STOP negative return. Therefore
-- StopReturn(StandardReturnHeader): result: negative: diagnostic:
-- diagnosticsExtension
-- (see CCSDS-CSTS-COMMON-OPERATIONS-PDUS module in reference [1]) is set
-- to ‘notUsed’.

-- *****
-- TRANSFER-DATA invocation extension parameters
-- No extension parameters added to the TRANSFER-DATA invocation.
-- The Cyclic Report procedure specified in reference [1] does not extend
-- the TRANSFER-DATA invocation. Therefore,
-- TransferDataInvocation: transferDataInvocationExtension
-- (see CCSDS-CSTS-COMMON-OPERATIONS-PDUS module in reference [1]) is set
-- to ‘notUsed’

-- TRANSFER-DATA invocation data parameter refinement
-- The Cyclic Report procedure specified in reference [1] resolves the data
-- parameter of the TRANSFER-DATA invocation to be of type
-- CyclicReportTransferDataRef. For the On-Change-Option Cyclic Report
-- procedure
-- TransferDataInvocation: data
-- is cast as
-- CyclicReportTransferDataRef: qualifiedParameters
-- (see the CCSDS-CSTS-CYCLIC-REPORT-PDUS module in reference [1]).

END
```

C2 TRANSFER SYNTAX

The `OnChangeOptCyclicReportStartInvocExt` type specified in this module shall be encoded for transfer using the Basic Encoding Rules specified in reference [G2].

ANNEX D

MONITORED DATA PRODUCTION

(NORMATIVE)

D1 GENERAL

D1.1 During the execution of a service package, the Functional Resources that are instantiated by that service package supply two kinds of data for use by MD-CSTS instances:

a) the current values of all functional resource monitored parameters that are collected and made available for use by that spaceflight mission; and

b) event notifications for all notifiable events that are made available for use by that spaceflight mission.

D1.2 The Monitored Data Collection function collects these monitored parameter values and notifiable events from the Functional Resources that comprise the service package, and formats them for transference by the On-Change Option Cyclic Report procedure instances, Information Query procedure instances, and Notification procedure instances of all MD-CSTS instances operating as part of the service package.

D2 MONITORED DATA COLLECTION FUNCTION BEHAVIOR

D2.1 During the execution of a service package, the Monitored Data Collection function shall collect the current values of all functional resource monitored parameters produced by (or an behalf of) all production Functional Resource instances in that service package.

D2.2 The Monitored Data Collection function shall format the value of each functional resource monitored parameter as of type `QualifiedParameter`, as specified in F4.3 of reference [1].

D2.3 The Monitored Data Collection function shall provide a given functional resource monitored parameter to every MD-CSTS instance in the service package that executes one or more instances of the On-Change-Option Cyclic Report procedure that subscribe to that parameter.

NOTE – How the Monitored Data Collection function identifies which MD-CSTS instances are subscribed to a given monitored parameter is an implementation detail that is outside the scope of this specification.

D2.4 The Monitored Data Collection function shall retain the value of each functional resource monitored parameter until a new value for that parameter is available. During the execution of a service package, the Monitored Data Collection function shall receive all

event notifications produced by (or on behalf of) all Functional Resource instances in that service package.

D2.5 Upon the occurrence of a given notifiable event, the Monitored Data Collection function shall provide the corresponding event notification to every MD-CSTS instance in the service package that executes one or more instances of the Notification procedure that subscribe to that event.

NOTE – How the Monitored Data Collection function identifies which MD-CSTS instances are subscribed to a given notifiable event is an implementation detail that is outside the scope of this specification.

D2.6 The Monitored Data Collection function shall format each event notification as an [eventName: eventValue] pair, where eventName has the type `Name` as specified in F4.3 of reference [1], and eventValue has the type `EventValue` as specified in F4.3 of reference [1].

D3 MONITORED DATA COLLECTION FUNCTION SERVICE MANAGEMENT INFORMATION

There is no standard set of explicit Service Management information associated with the Monitored Data Collection function. Every instance of the Monitored Data Collection function is assumed to have access to all monitored parameters and all notifiable events of all Functional Resources that execute as part of the service package to which the Monitored Data Collection function belongs.

D4 MONITORED DATA COLLECTION FUNCTIONAL RESOURCE TYPE

D4.1 The OID for the Functional Resource Type that represents the Monitored Data Collection function shall be as specified in the SANA Functional Resource Registry (reference [3]), using the FR classifier `MonitoredDataCollection`.

D4.2 Any monitored parameters, notifiable events, or directives that may be defined for the Monitored Data Collection FR type shall be registered under the `mdCollectionParametersId` ({`monitoredDataCollection` 1}), `mdCollectionEventsId` ({`monitoredDataCollection` 2}), or `mdCollectionDirectivesId` ({`monitoredDataCollection` 3}) nodes, respectively, of the SANA FR Registry.

ANNEX E

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

E1 SECURITY CONSIDERATIONS

E1.1 INTRODUCTION

This subsection describes security aspects of the Monitored Data service.

The CSTS Specification Framework Recommended Standard (reference [1]) explicitly provides authentication and access control for CSTSes. As one of the suite of CSTSes, the Monitored Data service inherits the authentication and access control capabilities defined in the CSTS Specification Framework Recommended Standard. The Monitored Data service provides no service-specific security capabilities. As specified in the CSTS Specification Framework, additional security capabilities, if required, are levied on the underlying communications services that support the MD-CSTS. Specification of the various underlying communications technologies, and in particular their associated security provisions, are outside the scope of this Recommended Standard.

E1.2 SECURITY CONCERNS WITH RESPECT TO THE MONITORED DATA SERVICE

The ‘Security Aspects of Cross Support Transfer Services’ subsection (H1 of reference [1]) identifies the support for capabilities that respond to security concerns in the areas of data privacy (also known as confidentiality), data integrity, authentication, access control, availability of resources, and auditing.

E1.3 POTENTIAL THREATS AND ATTACK SCENARIOS

As a member of the suite of CSTSes, the Monitored Data service depend on unspecified mechanisms operating in the underlying communications service, or on privacy-ensuring capabilities in the service-specific application processes that interoperate through the Framework procedures, to ensure data privacy (confidentiality). If no such mechanisms are actually implemented, or the mechanisms selected are inadequate or inappropriate to the network environment in which the mission is operating, an attacker could read the data contained in the MD-CSTS protocol data units as they traverse the WAN between service user and service provider.

The CSTS Specification Framework Recommended Standard (reference [1]) constrains the ability of a third party to seize control of an active CSTS instance, but it does not specify mechanisms that would prevent an attacker from intercepting the protocol data units.

Interception of monitored parameters such as the azimuth and elevation of the ground station antenna and the actual receive and transmit frequencies could assist an attacker in acquiring return link data or jamming the forward link. The prevention of such interception attacks depends on unspecified mechanisms in the underlying communications service. If no such mechanisms are actually implemented, or the mechanisms selected are inadequate or inappropriate to the network environment in which the mission is operating, an attacker could intercept data transferred between the service user and the service provider without detection.

If the CSTS authentication capability is not used and if authentication is not ensured by the underlying communications service, attackers could somehow obtain valid `initiator-identifier` values and use them to initiate MD-CSTS instances by which they could gain access to the data transferred via the service.

The MD-CSTS depends on unspecified mechanisms operating in the underlying communications service to ensure that the supporting network has sufficient resources to provide sufficient support to legitimate service users. If no such mechanisms are actually implemented, or the mechanisms selected are inadequate or inappropriate to the network environment in which the mission is operating, an attacker could prevent legitimate service users from using the MD-CSTS.

If the service provider of the MD-CSTS provides no security auditing capabilities, or if a service user chooses not to employ auditing capabilities that do exist, then attackers may delay or escape detection while stealing data exchanged via the service.

E1.4 CONSEQUENCES OF NOT APPLYING SECURITY TO THE TECHNOLOGY

The consequences of not applying security to the MD-CSTS are possible degradation and loss of ability to use the service, or the interception of data that provides information that could aid in the acquisition and/or jamming of the space link itself.

E2 SANA CONSIDERATIONS

The MD-CSTS relies on the SANA Functional Resource Registry (reference [3]) to provide the identification and definition of Functional Resource parameters and events;

As described in this Recommended Standard, the MD-CSTS reports parameters and events that are named in the context of Functional Resources. Functional Resource Types are registered under the

```
{  iso(1) identified-organization(3) standards-producing-organization(112)
   ccsds(4) css(4) crossSupportResources(2)
}
```

node of the OID registration tree.

There are two subnodes under the `crossSupportResources` node: `crossSupportFunctionalities` and `agencyFunctionalities`, used to register CCSDS-standard Functional Resource Types and agency-unique Functional Resource Types, respectively. Under each Functional Resource Type OID, the parameters, events, and directives are registered under dedicated subnodes.

Maintenance of the SANA registry of the Functional Resource Types, parameters, events, and directives under the `crossSupportFunctionalities` subnode is under the purview of the CCSDS Cross Support Services Area in accordance with the process and procedures identified in the CSTS Specification Framework Recommended Standard (reference [1]).

Maintenance of the SANA registry of the Functional Resource Types, parameters, events, and directives under the `agencyFunctionalities` subnode is under the purview of designated Agency-level control authorities. The process and procedure for designating Agency-level control authorities is documented in the CSTS Specification Framework Recommended Standard (reference [1]).

E3 PATENT CONSIDERATIONS

There are no patents that are known to apply to the technology used in the Monitored Data service.

ANNEX F

EXAMPLE FUNCTIONAL RESOURCE TYPE OBJECT IDENTIFIER REGISTRY

(INFORMATIVE)

F1 INTRODUCTION

This annex provides an example registry of Object Identifiers for the Functional Resource Types that are used in the operational scenario for the MD-CSTS (see 2.5). This example set of Object Identifiers has been modeled on the SANA Functional Resource Registry (reference [3]).

These examples are included in this informative annex to provide example Object Identifiers for the Functional Resource Types, parameters, and events that are cited abstractly in the operational scenario.

The SANA Functional Resource Registry is the official repository of all object identification assignments for Functional Resource Types and the parameters and notifiable events that belong to those Types. The example Object Identifiers documented in this annex do not necessarily correspond to those in the SANA registry, because this annex was written before the SANA registry was made final. Any real implementation of the MD-CSTS shall always use only the Functional Resource Types, parameters, and notifiable events that are specified in the SANA registry.

The Functional Resource Types used in the operational scenario section of this Recommended Standard are:

- Antenna;
- Forward Space Link Carrier Transmission;
- Forward TC PLOP, Sync, and Channel Encoding;
- Forward CLTU Transfer Service Provider;
- Return Space Link Carrier Reception;
- Return TM Synchronization and Decoding;
- Return All Frames Transfer Service Provider;
- Monitored Data CSTS Provider; and
- Monitored Data Collection.

As specified in D3 of reference [1], all Functional Resource Types are registered under one of two subnodes under `crossSupportResources` node of the OID registration tree:

```
{  iso(1) identified-organization(3) standards-producing-organization(112)
   ccsds(4) css(4) crossSupportResources(2)}
```

The `crossSupportFunctionalities` subnode is used to register CCSDS-standard Functional Resource Types, and the `agenciesFunctionalities` subnode is used to register Agency-specific Functional Resource Types. All Functional Resource Types listed above are registered under the `crossSupportFunctionalities` subnode:

```
{crossSupportResources crossSupportFunctionalities (1)}
```

Thus the OID for the `crossSupportFunctionalities` subnode is 1/3/112/4/4/2/1.

The `crossSupportFunctionalities` subnode is the root node for all Functional Resource Types described in this annex. Each of the Functional Resource Types described in this annex are registered as nodes directly under this root node.

As specified in reference [1], the parameters, notifiable events, and directives that are specific to each Functional Resource Type are registered under `parametersId`, `eventsId`, and `directivesId` subnodes under that Functional Resource Type's node. For the Functional Resource Types listed above, all have functional resource type-specific parameters, some have functional resource type-specific events, and some have functional resource type-specific directives which however are not relevant in the context of this Recommended Standard.

The following subsections identify the Object Identifiers associated with each of the Functional Resource Types listed above. Each subsection has a table identifying the classifiers, descriptions, types and OIDs of the functional resource type-specific parameters for that Functional Resource Type that are used in the operational scenario in 2.5. The full set of parameters for the actual set of Functional Resource Types are found in the SANA registry.

NOTE – In the following tables, parameter classifiers in the Parameter columns may contain hyphens. **These hyphens are not part of the parameter classifiers**: they are inserted merely for the purpose of allowing long parameter classifiers to break cleanly into the narrow Parameter columns. For example, 'antPoint-ingMode' in the Parameter column refers to the parameter with the classifier 'antPointingMode'.

If the Functional Resource Type has functional resource type-specific events that are used in the operational scenario, the subsection contains a table identifying the names, descriptions, types and OIDs of each functional resource type-specific event for that Functional Resource Type that is used in the scenario.

In accordance with the CSTS conventions for forming parameter and event OIDs, the last component of each OID is the version number. The version number for all parameter and event OIDs in the annex is ‘1’.

F2 ANTENNA

The Antenna Functional Resource Type is the first node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities antenna (1)}
```

Thus the OID for the Antenna Functional Resource Type is (1/3/112/4/4/2/1/1).

The Antenna `parametersId` node is the first subnode of the `antenna` node. Thus the OID for the Antenna `parametersId` node is (1/3/112/4/4/2/1/1/1).

Table F-1 identifies the parameters of the Antenna Functional Resource Type used in the operational scenario and provides their descriptions, types, and OIDs (registered under the Antenna `parametersId` node).

The Antenna `eventsId` node is the second subnode of the `antenna` node. Thus the OID for the Antenna `eventsId` node is (1/3/112/4/4/2/1/1/2).

The Antenna Functional Resource Type has no functional resource type-specific events that were used in the operational scenario.

Table F-1: Antenna Functional Resource Type Parameters

Parameter	Description	Type and Engineering Units	OID
antPointingMode	This enumerated parameter reports the pointing mode of the antenna servo system. The values this parameter can take on are: - 'stow' – the antenna is in or is moving to its stow position; - 'halt' – the antenna has been stopped in its current position; - 'slew' – the antenna is moving at commanded angular rates; - 'program-track' – the antenna is pointed in accordance with spacecraft trajectory predicts; - 'conical scan' – the antenna is performing a conical scan around the nominal pointing and applies offsets with respect to the predicts such that the observed signal strength is constant throughout the scan; - 'auto-track' – the antenna pointing is driven by a tracking receiver that by means of a suitable feed (e.g., monopulse) determines an error signal both for azimuth and elevation. Antenna implementations will typically support only a subset of the above listed pointing modes.	Type: SEQUENCE (SIZE (1)) OF INTEGER { stow (0) , halt (1) , slew (2) , programTrack (3) , conicalScan (4) , autoTrack (5) } Engineering Units: None	1/3/112/4/4/2/1/1/1/13/1

F3 FORWARD 401 SPACE LINK CARRIER TRANSMISSION

The Forward 401 Space Link Carrier Transmission Functional Resource Type is the second node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities fwd401SpaceLinkCarrierXmit (2)}
```

Thus the OID for the Forward Space Link Carrier Transmission Functional Resource Type is (1/3/112/4/4/2/1/2).

The Forward 401 Space Link Carrier Transmission `parametersId` node is the first subnode of the `fwd401SpaceLinkCarrierXmit` node. Thus the OID for the Forward Space Link Carrier Transmission `parametersId` node is (1/3/112/4/4/2/1/2/1).

Table F-2 identifies the parameters of the Forward 401 Space Link Carrier Transmission Functional Resource Type used in the operational scenario and provides their descriptions, types, and OIDs (registered under the Forward Space Link Carrier Transmission `parametersId` node).

The Forward 401 Space Link Carrier Transmission `eventsId` node is the second subnode of the `fwd401SpaceLinkCarrierXmit` node. Thus the OID for the Forward Space Link Carrier Transmission `eventsId` node is (1/3/112/4/4/2/1/2/2).

The Forward 401 Space Link Carrier Transmission Functional Resource Type has no functional resource type-specific events used in the operational scenario.

Table F-2: Forward 401 Space Link Carrier Transmission Functional Resource Type Parameters

Parameter	Description	Type and Engineering Units	OID
`fwd401-CarrierXmit-Actual-CarrierFreq`	This parameter reports the current forward link frequency in Hz. In general the frequency will be constant, except during the forward link sweep and for Category B missions in case the forward link is being ramped as to compensate for the Doppler shift and rate on the forward link.	Type: SEQUENCE (SIZE (1)) OF IntPos (2025000000 .. 40500000000) Engineering Units: Hz	1/3/112/4/4/2/1/2/1/5/1

F4 FORWARD TC PLOP, SYNC, AND CHANNEL ENCODING

The Forward TC PLOP, Sync, and Channel Encoding Functional Resource Type is the fourth node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities fwdTcPlopSyncAndChnlEncoding (4)}
```

Thus the OID for the Forward TC PLOP, Sync, and Channel Encoding Functional Resource Type is (1/3/112/4/4/2/1/4).

The Forward TC PLOP, Sync, and Channel Encoding `parametersId` node is the first subnode of the `fwdTcPlopSyncChnlAndEncoding` node. Thus the OID for the Forward TC PLOP, Sync, and Channel Encoding `parametersId` node is (1/3/112/4/4/2/1/4/1).

The Forward TC PLOP, Sync, and Channel Encoding Functional Resource Type has no monitored parameters that were used in the operational scenario.

The Forward TC PLOP, Sync, and Channel Encoding `eventsId` node is the second subnode of the `fwdTcPlopSyncAndChnlEncoding` node. Thus the OID for the Forward TC PLOP, Sync, and Channel Encoding `eventsId` node is (1/3/112/4/4/2/1/4/2).

The Forward TC PLOP, Sync, and Channel Encoding Functional Resource Type has no functional resource type-specific events that were used in the operational scenario.

F5 FORWARD CLTU TRANSFER SERVICE PROVIDER

The Forward CLTU Transfer Service (TS) Provider Functional Resource Type is the fourteenth node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities fwdCltuTsProvider (14)}
```

Thus the OID for the Forward CLTU TS Provider Functional Resource Type is (1/3/112/4/4/2/1/14).

The Forward CLTU TS Provider `parametersId` node is the first subnode of the `fwdCltuTsProvider` node. Thus the OID for the Forward CLTU TS Provider `parametersId` node is 1/(3/112/4/4/2/1/14/1).

Table F-3 identifies the parameters of the Forward CLTU TS Provider Functional Resource Type used in the operational scenario and provides their descriptions, types, and OIDs (registered under the Forward CLTU TS Provider `parametersId` node).

The Forward CLTU TS Provider `eventsId` node is the second subnode of the `forwardCltuTsProvider` node. Thus the OID for the Forward CLTU TS Provider `eventsId` node is (1/3/112/4/4/2/1/14/2).

Table F-4 identifies the events of the Forward CLTU TS Provider Functional Resource Type used in the operational scenario and provides their descriptions and OIDs (registered under the Forward CLTU TS Provider `eventsId` node).

Table F-3: Forward CLTU TS Provider Functional Resource Type Parameters

Parameter	Description	Type and Engineering Units	OID
`fwdCltu-TsSi-State`	This enumerated parameter reports the status of the given instance of the Forward CLTU service. It can take on the following values: - 'unbound' – all resources required to enable the provision of the Forward CLTU service have been allocated, and all objects required to provide the service have been instantiated; However, no association yet exists between the user and the provider; i.e., the Forward CLTU transfer service provider port is not bound; - 'ready'– an association has been established between the user and the provider, and they may interact by means of the service operations. However, sending of CLTUs from the user to the provider (by means of the CLTU-TRANSFER-DATA operation) is not permitted; the user may enable the delivery of CLTUs by means of the appropriate service operation (CLTU-START), which, in turn, will cause the provider to transition to the state 'active'; - 'active' – this state resembles state 'ready', except that now the user can send CLTUs and the provider is enabled to radiate CLTUs to the spacecraft; the service continues in this state until the user invokes the CLTU-STOP operation to cause the provider to suspend transmission of CLTUs and transition back to state 'ready' or the PEER-ABORT invocation to cause the service to transition back to the 'unbound' state.	Type: SEQUENCE (SIZE (1)) OF INTEGER { unbound (0) , ready (1) , active (2) } Engineering Units: None	1/3/112/4/4/2/1/14/1/6/1
`fwdCltu-Ts-Number-OfCltus-Radiated`	This parameter reports the number of CLTUs that the provider successfully radiated completely during the service provision period. A CLTU in the process of being radiated is not included in this count.	Type: SEQUENCE (SIZE (1)) OF IntUnsigned Engineering Units: None	1/3/112/4/4/2/1/14/1/15/1

Table F-4: Forward CLTU TS Provider Functional Resource Type Events

Event	Description	Event Value Name, Type, and Engineering Units	OID
`fwdCltuTs-ProdStat-Change`	This event notifies any change of the `fwdCltuProdStat` parameter.	Event Value Name: `fwdCltuProdStatValue` Type: `SEQUENCE (SIZE (1)) OF INTEGER { configured (0) , operational (1) , interrupted (2) , halted (3) }` Engineering Units: None	1/3/112/4/4/2/1/14/2/1/1

F6 RETURN 401 SPACE LINK CARRIER RECEPTION

The Return 401 Space Link Carrier Reception Functional Resource Type is the twenty-first node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities rtn401SpaceLinkCarrierRecpt (21)}
```

Thus the OID for the Return 401 Space Link Carrier Reception Functional Resource Type is (1/3/112/4/4/2/1/21).

The Return 401 Space Link Carrier Reception `parametersId` node is the first subnode of the `rtn401SpaceLinkCarrierRecpt` node. Thus the OID for the Return 401 Space Link Carrier Reception `parametersId` node is (1/3/112/4/4/2/1/21/1).

Table F-5 identifies the parameters of the Return 401 Space Link Carrier Reception Functional Resource Type used in the operational scenario and provides their descriptions, types, and OIDs (registered under the Return 401 Space Link Carrier Reception `parametersId` node).

Table F-5: Return 401 Space Link Carrier Reception Functional Resource Type Parameters

Parameter	Description	Type	OID
rtn401-CarrierRecpt-ActualFreq	This parameter reports the observed carrier return link frequency in Hz. This parameter therefore varies with the Doppler shift induced by the radial velocity of the spacecraft relative to the ground based antenna. In 1-way mode, the Doppler shift applies only once, but also the onboard oscillator drift affects the observed return link carrier frequency. In 2-way mode in combination with a constant forward link frequency, the Doppler shift approximately doubles with respect to the 1-way case, but the contribution of the onboard oscillator drift is eliminated.	Type: SEQUENCE (SIZE (1)) OF IntUnsigned (2199700000 .. 38500000000) Engineering Units: Hz	1/3/112/4/4/2/1/21/1/15/1
rtn401-CarrierRecpt-Subcarrier-LockStat	This enumerated parameter reports on the subcarrier lock status of the BPSK subcarrier demodulator. The values the parameter may have are: - 'locked' – the demodulator has locked on the return link subcarrier; - 'not-locked' – the demodulator has not locked on the return link subcarrier and therefore cannot deliver telemetry. If the applicable modulation scheme does not use a subcarrier, this parameter shall be flagged as undefined.	Type: SEQUENCE (SIZE (1)) OF INTEGER { locked (0) , not-locked (1) } Engineering Units: None	1/3/112/4/4/2/1/21/1/21/1
rtn401-CarrierRecpt-Actual-Subcarrier-Freq	This parameter reports the actually received subcarrier frequency in 1/1000 Hz; i.e., this parameter reflects the Doppler shift of the subcarrier frequency. If the applicable modulation scheme does not use a subcarrier, this parameter shall be flagged as undefined.	Type SEQUENCE (SIZE (1)) OF IntUnsigned (2000 .. 300000) Engineering Units: 1/1000 Hz	1/3/112/4/4/2/1/21/1/23/1
rtn401-CarrierRecpt-Symbol-SyncLockStat	This enumerated parameter reports the symbol synchronizer lock status. The values the parameter may have are: - 'locked' – the symbol synchronizer has locked on the return link symbol stream; - 'not-locked' – the symbol synchronizer has not locked on the symbol stream.	Type SEQUENCE (SIZE (1)) OF INTEGER { locked (0) , not-locked (1) } Engineering Units: None	1/3/112/4/4/2/1/21/1/25/1

The Return 401 Space Link Carrier Reception `eventsId` node is the second subnode of the `rtn401SpaceLinkCarrierRecpt` node: thus the OID for the Return 401 Space Link Carrier Reception `eventsId` node is (1/3/112/4/4/2/1/21/2).

The Return 401 Space Link Carrier Reception Functional Resource Type has no functional resource type-specific events that were used in the operational scenario.

F7 RETURN TM SYNCHRONIZATION AND CHANNEL DECODING

The Return Telemetry (TM) Synchronization and Channel Decoding Functional Resource Type is the twenty-third node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities rtnTmSyncAndChnlDecoding (23)}
```

Thus the OID for the Return TM Synchronization and Channel Decoding Functional Resource Type is (3/112/4/4/2/1/23).

The Return TM Synchronization and Channel Decoding `parametersId` node is the first subnode of the `rtnTmSyncAndChnlDecoding` node. Thus the OID for the Return TM Synchronization and Channel Decoding `parametersId` node is (3/112/4/4/2/1/23/1).

The Return TM Synchronization and Channel Decoding Functional Resource Type has no functional resource type-specific parameters that were used in the operational scenario.

The Return TM Synchronization and Decoding `eventsId` node is the second subnode of the `rtnTmSyncAndChnlDecoding` node. Thus the OID for the Return TM Synchronization and Decoding `eventsId` node is (3/112/4/4/2/1/23/2).

Table F-6 identifies the events of the Return TM Synchronization and Channel Decoding Functional Resource Type used in the operational scenario and provides their descriptions and OIDs (registered under the Return TM Synchronization and Decoding `eventsId` node).

Table F-6: Return TM Synchronization and Channel Decoding Functional Resource Type Events

Event	Description	Event Value Name, Type, and Engineering Units	OID

`rtnTmSync-FrameSync-LockStat-Change`	This event notifies the successful acquisition of frame lock; i.e., in two consecutive frames the ASM was detected at the expected position and with a correlation error not exceeding the permitted limit.	Event Value Name: rtnTmSyncFrameSyncLockStat-Value
		
		Type:
		`SEQUENCE (SIZE (1)) OF INTEGER`
		`{ locked (0)`
		`, verify (1)`
		`, notLocked (2)`
		`}`
		
		Engineering Units: None
			1/3/112/4/4/2./1/23/2/1/1

F8 RETURN ALL FRAMES TRANSFER SERVICE PROVIDER

The Return All Frames Transfer Service (TS) Provider Functional Resource Type is the twenty-ninth node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities rtnAfTsProvider (29)}
```

Thus the OID for the Return All Frames TS Provider Functional Resource Type is (3/112/4/4/2/1/29).

The Return All Frames TS Provider `parametersId` node is the first subnode of the `rtnAfTsProvider` node. Thus the OID for the Return All Frames TS Provider `parametersId` node is (3/112/4/4/2/1/29/1).

Table F-7 identifies the parameters of the Return All Frames TS Provider Functional Resource Type used in the operational scenario and provides their descriptions, types, and OIDs (registered under the Return All Frames TS Provider `parametersId` node).

The Return All Frames TS Provider `eventsId` node is the second subnode of the `rtnAfTsProvider` node. Thus the OID for the Return All Frames TS Provider `eventsId` node is (3/112/4/4/2/1/29/2).

Table F-8 identifies the events of the Return All Frames TS Provider Functional Resource Type used in the operational scenario and provides their description sand OIDs (registered under the Return All Frames TS Provider `eventsId` node).

Table F-7: Return All Frames TS Provider Functional Resource Type Parameters

Parameter	Description	Type and Engineering Units	OID
rtnAfSi-State	This enumerated parameter reports the status of the given instance of the Return All Frames service. It can take on the following values: - 'unbound' – All resources required to enable the provision of the Return All Frames service have been allocated, and all objects required to provide the service have been instantiated; however, no association yet exists between the user and the provider; i.e., the Return All Frames transfer service provider port is not bound; - 'ready'– An association has been established between the user and the provider, and they may interact by means of the service operations. However, sending of telemetry frames from the provider to the user (by means of the RAF-TRANSFER-DATA operation) is not permitted; the user may enable the delivery of telemetry frames by means of the appropriate service operation (RAF-START), which, in turn, will cause the provider to transition to the state 'active'; - 'active' – This state resembles state 'ready', except that now the provider will send telemetry frames provided frames of the selected characteristics are made available by the RAF production process; the service continues in this state until the user invokes the RAF-STOP operation to cause the provider to suspend delivery of telemetry frames and transition back to state 'ready' or the PEER-ABORT invocation to cause the service to transition back to the 'unbound' state.	Type: SEQUENCE (SIZE (1)) OF INTEGER { unbound (0) , ready (1) , active (2) } Engineering Units: None	1/3/112/4/4/2/1/29/1/7/1
rtnAf-NumberOf-Frames-Delivered	This parameter reports the total number of telemetry frames that were delivered to the user since the start of the service instance provision period.	Type: SEQUENCE (SIZE (1)) OF IntUnsigned Engineering Units: None	1/3/112/4/4/2/1/29/1/12/1

Table F-8: Return All Frames TS Provider Functional Resource Type Events

Event	Description	Event Value Name, Type, and Engineering Units	OID
rtnAfProd-StatChange	This event notifies any change of the rtnAfProdStat parameter.	Event Value Name: rtnAfProdStatValue Type: SEQUENCE (SIZE (1)) OF INTEGER { running (0) , interrupted (1) , halted (2) } Engineering Units: None	1/3/112/4/4/2/1/29/2/1/1

F9 MD-CSTS PROVIDER

The MD-CSTS Provider Functional Resource Type is the forty-sixth node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities mdCstsProvider (46)}
```

Thus the OID for the MD-CSTS Provider Functional Resource Type is (3/112/4/4/2/1/46).

The MD-CSTS Provider `parametersId` node is the first subnode of the `mdCstsProvider` node. Thus the OID for the MD-CSTS Provider `parametersId` node is (3/112/4/4/2/1/46/1).

Table F-9 identifies the parameter of the MD-CSTS Provider Functional Resource Type that is used in the operational scenario and provides its description, type, and OID (registered under the MD-CSTS Provider `parametersId` node).

Table F-9: MD-CSTS Provider Functional Resource Type Parameters

Parameter	Description	Type and Engineering Units	OID
mdNamed-Label-Lists	This parameter contains the set of Label Lists that may be used for all instance of the On-Change-Option Cyclic Report and Information Query procedures.	Type: LabelListSet as defined in F4.16 of reference [1] Engineering Units: None	3/112/4/4/2/1/46/1/1/1

The MD-CSTS Provider Functional Resource Type has one CSTS *service-generic* event that is used in the operational scenario.

The CSTS Specification Framework Recommended Standard (reference [1]) defines service-generic parameters and events that are available for use by CSTSes, which in Functional Resource terms means that these service-generic parameters and events are available to the MD-CSTS via the CSTS Provider Functional Resource Types for those CSTSes. The OIDs for these CSTS Provider service-generic parameters and events are registered under the `serviceGenericIdentifiers` subnode of the (CSTS) `framework` node:

```
{  iso(1) identified-organization (3) standards-producing-organization(112)
   ccsds(4) css(4) csts(1) framework(1) serviceGenericIdentifiers(5)
   serviceGenericId(1)
}
```

The service-generic functional resource parameters are registered under the `svcParametersId` subnode of `serviceGenericIdentifiers`:

```
{serviceGenericId svcParametersIdentifiers (1)}
```

The service-generic functional resource events are registered under the `svcEventsId` subnode of `serviceGenericIdentifiers`:

```
{serviceGenericId svcEventsIdentifiers (2)}
```

The description and OID of this service-generic event can be found in 3.11.2.2.3.2 and F4.17, respectively, of reference [1].

The MD-CSTS Provider Functional Resource Type assigns the OID and semantics of the `svcProductionStatusChangeVersion1` service-generic functional resource event to the MD-CSTS Provider FR-specific event with the classifier `mdCstsProdStatChange`.

Table F-10 identifies the event of the MD-CSTS Provider Functional Resource Type used in the operational scenario and provides its description, type, and OID.

Table F-10: MD-CSTS Provider Functional Resource Type Events

Event	Description	Event Value Name, Type, and Engineering Units	OID
`mdCsts-ProdStat-Change`	This event notifies any change of the `mdCstsProdStat` parameter.	Event Value Name: `mdCstsProdStatValue` Type: `SvcProductionStatus-Version1` as defined in F4.17 of reference [1] Engineering Units: None	The OID value of the `svcProduction-StatusChange-Version1` parameter defined in in F4.17 of reference [1]

F10 MONITORED DATA COLLECTION

The Monitored Data Collection Resource Type is the forty-seventh node under the `crossSupportFunctionalities` node:

```
{crossSupportFunctionalities monitoredDataCollection (47)}
```

Thus the OID for the Monitored Data Collection Functional Resource Type is (3/112/4/4/2/1/47).

The Monitored Data Collection Functional Resource Type has no functional resource type-specific parameters or events that are used in the operational scenario.

ANNEX G

INFORMATIVE REFERENCES

(INFORMATIVE)

[G1] *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.

[G2] *Space Link Extension—Internet Protocol for Transfer Services*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 913.1-B-1. Washington, D.C.: CCSDS, September 2008.

[G3] *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.

[G4] *Space Link Extension—Forward CLTU Service Specification*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 912.1-B-3. Washington, D.C.: CCSDS, July 2010.

[G5] *Space Link Extension—Return All Frames Service Specification*. Issue 3. Recommendation for Space Data System Standards (Blue Book), CCSDS 911.1-B-3. Washington, D.C.: CCSDS, January 2010.

[G6] *Cross Support Service Management—Service Management Utilization Request Formats*. Issue 1. Recommendation for (Blue Book), CCSDS 902.9-B-1. Washington, D.C.: CCSDS, forthcoming.

[G7] *Cross Support Service Management—Simple Schedule Format Specification*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.1-B-1. Washington, D.C.: CCSDS, forthcoming.

[G8] *Extensible Space Communication Cross Support—Service Management—Concept*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 902.0-G-1. Washington, D.C.: CCSDS, September 2014.

[G9] *Space Communications Cross Support—Architecture Description Document*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 901.0-G-1. Washington, D.C.: CCSDS, November 2013.

[G10] *Radio Frequency and Modulation Systems—Part 1: Earth Stations and Spacecraft*. Issue 26. Recommendation for Space Data System Standards (Blue Book), CCSDS 401.0-B-26. Washington, D.C.: CCSDS, October 2016.

[G11] *Data Transmission and PN Ranging for 2 GHz CDMA Link via Data Relay Satellite*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 415.1-B-1. Washington, D.C.: CCSDS, September 2011.

[G12] *TC Synchronization and Channel Coding*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 231.0-B-2. Washington, D.C.: CCSDS, September 2010.

ANNEX H

ACRONYMS

(INFORMATIVE)

ASN.1	Abstract Syntax Notation One
BPSK	binary phase shift keying
CCSDS	Consultative Committee for Space Data Systems
CLTU	Communication Link Transmission Unit
CM	Complex Management
CSTS	Cross Support Transfer Service
ESLT	Earth Space Link Terminal
ISO	International Organization for Standardization
MD-CSTS	Monitored Data Cross Support Transfer Service
MDOS	Mission Data Operations System
MUE	Mission User Entity
OID	Object Identifier
PDU	protocol data unit
PLOP	Physical Layer Operations Procedure
QPSK	quadrature phase shift keying
RF	radio frequency
SANA	Space Assigned Numbers Authority
SLE	Space Link Extension
Sync	synchronization
TC	Telecommand
UM	Utilization Management

ANNEX I

CROSS REFERENCES TO CROSS SUPPORT TRANSFER SERVICE SPECIFICATION FRAMEWORK

(INFORMATIVE)

Table I-1 lists the specific sections and paragraphs of the Cross Support Transfer Service Specification Framework (reference [1]) that are referenced by this Recommended Standard, and identifies the sections and paragraphs of this Recommended Standard that make specific reference to each of those sections/paragraphs in reference [1].

Table I-1: Cross Reference to Reference-[1] Sections and Paragraphs

Reference-[1] Section/Paragraph	Referencing Sections/Paragraphs of MD-CSTS
3.4.2.2.4.3	7.1, 7.2
3.11.2.2.3.2 a)	8.1 b)
3.11.2.2.3.2 b)	8.1 c), 9.1 Note
3.11.2.2.3.2	F9
3.12.1.3 c)	5.4.1.1
3.12.1.3 d)	5.4.1.2
3.12.1.3 g)	5.4.1.3
3.12.1.4 a) 1)	5.4.1.4
3.12.1.4 b) 1)	5.4.1.5
3.12.1.4 c)	5.4.1.6
3.12.1.4 g)	5.4.1.7
3.12.2.4.1 c)	9.4.1
3.12.2.4.1 d)	9.4.2
3.12.2.4.1 g)	9.4.3
4.3	table 3-1, table A-5
4.3.5	7.3.1, 7.3.2, 7.3.3, 7.3.4
4.4	table A-5
4.9	table 3-1, table A-5
4.9.3.2	5.4.2
4.9.5	table 3-1, 7.6

Reference-[1] Section/Paragraph	Referencing Sections/Paragraphs of MD-CSTS
4.10	table 3-1, table A-5
4.10.3.1.3 c)	4.4.2.1
4.10.3.1.3 d)	4.4.2.2
4.10.3.1.3 e)	4.4.2.3
4.10.3.1.3 f)	4.4.2.4
4.10.3.1.3 g)	4.4.2.5
4.10.3.1.3 h)	4.4.2.6
4.10.3.2.3	4.4.3.1
4.10.3.2.4 a) 1)	4.4.3.2
4.10.3.2.4 b) 1)	4.4.3.3
4.10.3.2.4 c)	4.4.3.4
4.10.3.2.4 g)	4.4.3.5
4.10.4.1.3.1 c)	9.4.1
4.10.4.1.3.1 d)	9.4.2
4.10.4.1.3.1 g)	9.4.3
4.10.5	7.4.1, 7.4.3, 7.4.5
4.11	table 3-1, table A-5
4.11.3.1.3 c)	6.4.2
4.11.3.1.3 d)	6.4.3
4.11.3.1.3 e)	6.4.4
4.11.3.1.3 f)	6.4.5
4.11.3.1.3 g)	6.4.6
4.11.3.1.3 h)	6.4.7
4.11.3.2.2 a) 1)	6.4.8
4.11.3.2.2 b) 1)	6.4.9
4.11.3.2.2 c)	6.4.10
4.11.3.2.2 g)	6.4.11
4.11.4.1.3 c)	9.4.4
4.11.4.1.3 d)	9.4.5
4.11.4.1.3 g)	9.4.6

Reference-[1] Section/Paragraph	Referencing Sections/Paragraphs of MD-CSTS
4.11.5	7.5.1, 7.5.3
annex B	8.1
D3	F1
annex F	A1.2
F4.1	A2.2, table A-7, table A-10, table A-12, table A-14, table A-16, table A-18
F4.3	A2.2, table A-7, table A-8, table A-10, table A-11, table A-12, table A-13, table A-14, table A-15, table A-16, table A-17, table A-18, table A-19, D2.2, D2.6
F4.4	A2.2, table A-6, table A-12, table A-13, table A-14, table A-15, table A-16, table A-18, table A-19
F4.5	A2.2, table A-6, table A-7, table A-8, table A-9, table A-10
F4.12	A2.2, table A-14, table A-19
F4.13	A2.2, table A-14
F4.16	2.5.3.9.3 Note 1, table F-9
F4.17	8.1 a), 8.1 b), 8.1 c), 8.2, 8.3, 9.2, Note, 9.3, F9, table F-10
G3	3.3
H1	E1.2