TERMA
SPACE DIVISION

SLE Toolkit

Commonality Analysis Technical Note

Document No.: TERMA/SPD/63/MCS/SLE/DJF/TN/0001

Date: 19 Jan 2005

Issue: 1

Revision: 2

Author: S. R. Smith

Technical Review: G. Villemos

Standards Review: D. Nielsen

Authorised by: M. Alberti

Approved by: M. Alberti

Document Change Record

Issue	Date	Change
1.0	13 Dec 2004	Completed internal review.
1.1	07 Jan 2005	Updated following ESOC review
1.2	19 Jan 2005	Updated following ESOC review

Document Status Sheet

Issue	Page
1.0	All
1.1	All Covering RIDs ES001 – ES015
1.2	ALL Covering RIDs ES016 – ES020

Document Distribution

Company/Organisation	Name
ESOC	Y.Doat
	W. Hell
TERMA	M. Alberti
	G.Villemos
	S.R.Smith
	D.S. Nielsen

Table of Contents Page

Table of Figures

Table of Tables

1 Introduction

1.1 Purpose

This Technical Note (**TN**) documents the commonality analysis performed as part of the SLE Toolkit study.

The intended audience is the ESOC Technical Staff, technical staff of external agencies and the TERMA technical team.

1.2 Scope

This TN is the first of two outputs of the SLE toolkit WP 2.1, which considers the 'commonality' of the SLE API across the different SLE Services. The work package aims to review

> "…the CCSDS Recommendations ([CLTU-Pink], [FSP-Red], [RAF-Pink], [RCF-Red], [ROCF-Red]), identify all common operations and identify what are the commonalities between the required parameters." [SOW].

The latest versions of the CCSDS recommendations have been agreed to be the baseline of the project, i.e. the actual reviewed documents are [CLTU-Blue], [FSP-Blue], [RAF-Blue], [RCF-Blue] and [ROCF-Blue].

The commonality analysis covers operations of all services, parameters of the operations as well as the service instance state diagrams. This technical note thereby corresponds to the following Requirement Baseline [RB] requirements

SLE 1, 1.1, 1.2, 1.3, 1.4, 1.5.

BIND 1, 2.

OPER 1, 2, 3.

The actual SLE toolkit requirements are not within the scope of this document, but within the second TN created as part of WP 2.1.

Considerations regarding a generic CCSDS recommendation have already been made within the CCSDS group and a draft CCSDS recommendation has been created. This draft as well as other output of the CCSDS group's analysis has been included in this technical note.

1.3 Summary and Conclusion

In this Technical Note we have considered the commonality across Operations and Parameters as well as Service Provider State Transitions. We have concluded that:

- Operations and Parameters

  A commonality has been identified for the Operations and Parameters. This can be expressed in terms of a Generic Service Common Operation Structure that consists of a Header area and a Data area and is based on the concept of:

  - Service Common Data Types and Operation Structures
  - Service Specific Data Types and Operation Structures

  Basically the Header area contains Parameters that are of a Service Common Data Type, whereas the Data area can contain parameters of both Service Common Data Type (but not required by ALL services) and Service Specific Data Types.

  We have further refined this definition into Association Operations and Data Exchange Operations. The Association Operations consist of purely Operation Common Structures (entirely Headers and no Data area) and the Service Specific information is then relevant for the Data Exchange Operations.

  One final area that has been considered is a rationalisation/harmonisation of the parameter names. This has been proposed initially by considering the service parameter names for an individual operation but also in an attempt to have commonality across service operations. It must be noted though that this could have a significant impact on backward compatibility.

- Service Provider State Transitions

  It has not been possible to identify a commonality across all Services. However, the Return Services do appear to have a complete commonality that could be encapsulated within a common state machine.

1.4 Applicable Documents

The following documents are considered applicable to this Technical Note:

Item	Title	Reference	Issue	Date
[SOW]	SLE Toolkit – Statement of Work	SLE-10001-SOW-0001-TOS-GIB	1/0	July 2004
[RB]	SLE Toolkit Requirements Baseline	SLE-GSI-RB-10001-TOS-GIB	1/0	July 2004
[Prop-Tech]	SLE Toolkit Technical Proposal	ITT/RFQ: RFP-85	1.1	October 2004
[SDP]	SLE Toolkit Software Development Plan	ITT/RFQ: RFP-85	draft	October 2004
[RM]	Cross Support Reference Model Part 1: Space Link Extension Services. Recommendation for Space Data System Standards	CCSDS 910.4-B-1		May 1996

Item	Title	Reference	Issue	Date
[CLTU-Blue]	CCSDS Recommendation for SLE Forward CLTU service	CCSDS 912.1-B2		Nov 2004
[FSP-Blue]	CCSDS Recommendation for SLE Forward Space Packet service	CCSDS 912.3-B1		Nov 2004
[RAF-Blue]	CCSDS Recommendation for SLE Return All Frames service	CCSDS 911.1-B2		Nov 2004
[RCF-Blue]	CCSDS Recommendation for SLE Return Channel Frame service	CCSDS 911.2-B1		Nov 2004
[ROCF-Blue]	CCSDS Recommendation for SLE Return Operational Control Field service	CCSDS 911.5-B1		Nov 2004
[SLE-API]	SLE API specification	SLES-SW-API-0001-TOS-GCI	2.0	May 2004
	SLE API proxy specification,	SLES-SW-API-0002-TOS-GCI	1.2	May 2004
	SLE API ADD	SLES-ASW-ADD-1001-TOS-GCI	2.0,	May 2004
	SLE API Reference manual	SLES-ASW-RM-1001-TOS-GCI		
	SLE C++ API Supplement – Forward CLTU Service	SLES-SW-API-0201-TOS-GCI	2.0	May 2004
	SLE C++ API Supplement – Return All Frames Service	SLES-SW-API-0101-TOS-GCI	2.0	May 2004
	SLE C++ API Supplement – Return Channel Frames Service	SLES-SW-API-0102-TOS-GCI	2.0	May 2004
	SLE C++ API Supplement – FSP		2.0	May 2004
	SLE C++ API Supplement – R-OCF		2.0	May 2004
	SLE API system test plan	SLES-ASW-STP-0001-TOS-GCI		
	SLE API CIG (Installation Guide)			
[WN9923]	Applicable CCSDS recommendations for SLE transfer services	WN9923	1.1	28.01.2000

1.5 Reference Documents

Reference	Document
[OSI]	Information Technology – Open Systems Interconnection Basic Reference Model : The Basic Model. International Standard ISO/IEC 7498-1. 2nd ed. Geneva: ISO 1994

1.6 Abbreviations

Abbreviations used in this document:

CCSDS Colsultative Committee for Space Data Systems (http://www.ccsds.org/)

CLTU	Communications Link Transmission Unit
FSP	Forward Service Packet
PDU	Protocol Data Unit
RAF	Return All Frames
RCF	Return Channel Frames
ROCF	Return Operational Control Fields
SLE	Space Link Extension
TN	Technical Note

1.7 Terms and Definitions

The following terms and definitions are used within the document.

1.7.1 Space Link Extension Reference Model

SLE Services are specified within the framework defined by the SLE Reference Model (reference [RM]). The following subsections summarize selected concepts from the SLE reference model.

Term	Definition
Abstract Object	An abstract object is a functional entity that interacts with other abstract objects. Objects are of different types, which determine their function and behaviour. An object is characterized by its interfaces (one or more), which are called abstract ports, and the operations that are made available through those interfaces.
Abstract Service	An abstract service is the capability provided by a set of operations that an abstract object exposes at one or more of its abstract ports. NOTE – The concept of an abstract service is to be distinguished from the concept of an (N)-service as defined in the OSI Basic Reference Model (reference [OSI]). The definition of (N)-service is in terms of the capability provided by one layer in the OSI architecture to the layer above it. The definition of abstract service is in terms of the capability provided by one abstract object to another abstract object. In a cross support scenario, where one Agency is providing an SLE service to another Agency, the object that provides the service typically is associated with one Agency, and the object that uses the service typically is associated with the other Agency.
Abstract Binding	When two abstract ports have an association established between them, they are said to be bound. The act of establishing such an association is called abstract binding. One object (the initiator) invokes a bind operation which is accepted (or rejected) by another object (the responder).

Term	Definition
Service User/Provider	An object that offers a service to another by means of one or more of its ports is called a service provider (provider). The other object is called a service user (user). An object may be a provider of some services and a user of others. The terms user and provider are used to distinguish the roles of two interacting objects. In this Recommendation, when two objects are involved in provision of a service, the object closer to the space link is considered to be the provider of the service, and the object further from the space link is considered to be the user.
Operation	An operation is a procedure or task that one object (the invoker) can request of another (the performer) through a bound port pair. The terms invoker and performer are used to describe the interaction between two objects as the operations that constitute the service occur. One object invokes an operation that is performed by the other. For most services, each object invokes some operations and performs others.
Parameter	A parameter of an operation is data that may accompany the operation's invocation or return.
Data Type	The actual value of a parameter is constrained by a data type that is the set of possible values a parameter can have.
Protocol Data Unit	The complete set of parameters applicable to a particular operation, which are exchanged between Service Users and Providers.

1.8 Document Overview

This document contains the following chapters:

Chapter 1 defines the scope and purpose of the document and contains a summary of the document.

Chapter 2 provides the background and goal of the SLE toolkit study.

Chapter 3 contains the analysis of the existing recommendations.

Chapter 4 propose and defines the new SLE toolkit commonality concepts.

2 SLE Toolkit Background

The SLE Toolkit project is a study aimed at establishing the basis for a more generic approach to defining and implementing future SLE services.

Currently, the CCSDS SLE service recommendations define a complete set of operations for each service and this has resulted in a replication of common operations. An alternative approach is to use a “Toolkit”, where the common operations are grouped into a single recommendation, with service specific recommendations being defined as deltas in regards to the common recommendation, i.e. a building block principle.

The study will evaluate the Toolkit approach by analysis of the currently existing SLE Service recommendations, and demonstrate the approach by prototyping selected Toolkit operations and a dummy SLE Service making use of the Toolkit. The Toolkit envisages that the common operations are grouped in two layers within the SLE API

- An Association layer, encompassing all operations related to BIND
- An Operations layer, encompassing all other operations that a specific SLE Service could use.

The specifics of the actual SLE Service is handled by the Service Layer, which is built on the Toolkit layers, as shown below.

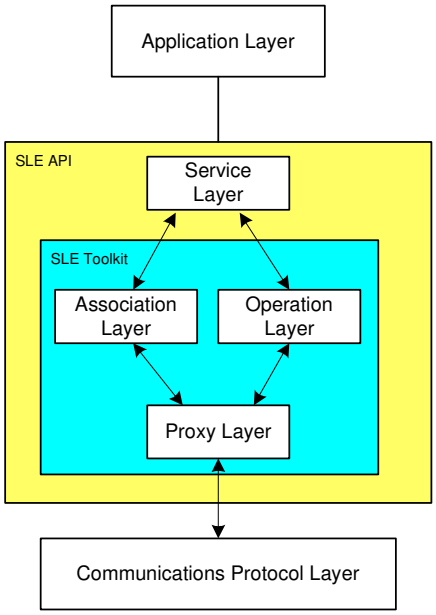

Figure 2-1: Toolkit Architecture

The project is organised in three main phases:

1. Analysis, covering the analysis of the current CCSDS SLE Service recommendations, analysis of the potential operations to be supported by the Toolkit, and analysis of the impact on the current SLE API software.
2. Design, covering production of draft CCSDS recommendation for the SLE Toolkit, and design of the SLE API incorporating the SLE Toolkit
3. Prototyping, covering the implementation and validation of the prototype SLE Toolkit and dummy SLE Service. This phase is completed by a presentation of the prototype, which also marks the completion of the work.

3 Commonality Analysis

This chapter provides the background information and the preliminary analysis of the SLE toolkit. The information is largely repetition from the Statement of Work [SOW] and the proposal [PROP], included for completeness and to provide a background reference for readers not familiar with the projects preliminary analysis.

3.1 Operations and Parameters

This section describes the operations and parameters of the SLE services. The section is subdivided into:

- A section defining the concept of operations.
- A section defining the concept of parameters.
- A section describing all association operations and their parameters.
- A section describing all data exchange operations and their parameters.

3.1.1 Operations

The aim of this part of the analysis is to identify and categorise common operations of the existing Return and Forward Services. In CCSDS Recommendation terms, an operation is defined as:

> *"A procedure or task that one object (the invoker) can request of another (the performer) through a bound port"*

This would appear to indicate that tasks related to establishing and releasing an association between two objects (through a bound port) are not strictly 'operations' in the above CCSDS terminology. However, in all CCSDS Recommendation Service documents, [CLTU-Blue], [FSP-Blue], [RAF-Blue], [RCF-Blue] and [ROCF-Blue], these tasks are still referred to as so-called operations. This naturally leads into a logical categorisation of the existing operations between a SLE Service User and a SLE Service Provider:

Association Operations - those related to establishing/releasing an association through a bound port

Service Operation	Invoked By	Purpose	Confirmed
BIND	User or Provider[1]	To establish an association between a User and Provider	Yes
UNBIND	User or Provider[1]	To release an association previously established by a BIND operation	Yes

[1] It should be noted that the ESA SLE API only supports User invoked BIND and UNBIND.

Service Operation	Invoked By	Purpose	Confirmed
PEER-ABORT	User or Provider	To notify the peer system that the local system detected an error that requires the association to be terminated	No

Table 3-1: Association Operations

As noted in the above table, the CCSDS Recommendations state that the BIND and UNBIND operations can be invoked by either User or Provider. However, the current ESA SLE API only supports User invoked BIND and UNBIND. It is not obvious that Provider invoked BIND/UNBIND operations bring any added capability to SLE Services and, if this constraint were removed to allow just User invoked BIND/UNBIND operations, whether any real functionality would be lost. It is clear that this change would lead to operational changes (procedural) when using SLE Services, but it could have a significant (positive) impact on the SLE API commonality, which is the purpose of this study. This issue would need to be discussed at a CCSDS Meeting for agreement/approval, as it is a change to the CCSDS Recommendations.

Data Exchange Operations - those relevant to the transfer of data through a bounded port

Service Operation	Invoked By	Purpose	Confirmed
START	User	To request that: for Forward Services, the SLE Service Provider accept TRANSFER-DATA operations from the User for Return Services, the SLE Service Provider sends TRANSFER-DATA operations to the User	Yes
STOP	User	To request that the SLE service provider stop service provision.	Yes
TRANSFER-DATA	User or Provider	To transfer data: for Forward Services, the SLE Service User sends data to the Provider for Return Services, the SLE Service Provider sends data to the User	Yes
SYNC-NOTIFY	Provider[2]	To notify the user of an event affecting production or provision of the service	No
ASYNC-NOTIFY			
SCHEDULE-STATUS-REPORT	User	To request that the provider send a status report immediately or periodically, or stop reporting	Yes
STATUS-REPORT	Provider	To send a status report to the user	No

[2] Notification Operations – Return Services use SYNC-NOTIFY whereas Forward Services use ASYNC-NOTIFY.

Service Operation	Invoked By	Purpose	Confirmed
GET-PARAMETER	User	To ascertain the value of an SLE service parameter	Yes
THROW-EVENT	User	To forward an event that requires Complex Management to take the actions defined for this event	Yes
INVOKE-DIRECTIVE[3]	User	To invoke TC-Directives required to (re-) establish the commanding capability on a given VC.	Yes

Table 3-2: Data Exchange Operations

[3] INVOKE-DIRECTIVE operation currently only applicable to FSP Service

3.1.2 Parameters

Associated with each operation are a number of parameters. In CCSDS Recommendation terms, a parameter is defined as:

> *"A parameter of an operation is data that may accompany the operation's invocation or return "*

The actual value of a parameter is constrained by an underlying data type, which is a set of possible values that the parameter can have. It should be noted however, that the values of many parameters are represented by names but the name of a value does not imply anything about its data type, eg.

Operation	BIND
Parameter Name	Diagnostic
Data Type/Names	BindDiagnostic

- AccessDenied
- ServiceTypeNotSupported
- VersionNotSupported
- NoSuchServiceInstance
- AlreadyBound
- SiNotAccessibleToThisInitiator
- InconsistentServiceType
- InvalidTime
- OutOfService
- otherReason

The set of parameters applicable to a particular operation are exchanged between User and Provider applications within a Protocol Data Unit (PDU).

The aim of this part of the analysis is to identify commonalities of the parameters required for the operations described in the previous section. If we review the Data Type Definitions for the existing services (defined in Annex A of the individual CCSDS Recommendation documents) we can see that they have been categorised as:

- Common Service Definitions
  - Basic types that are common with other SLE Transfer Services (Common Types)
  - Types exchanged between a User and Provider application in order to establish, release or abort an association (BIND Types)
  - Common invocation (I) and return (R) PDUs (Common PDUs)
  - Service Instance Identifiers

- Service Specific Definitions
  - Data types
  - Incoming PDUs (from the Provider point of view)
  - Outgoing PDUs (from the Provider point of view)

From the above we can see that these definitions cover both the 'Parameter Data Type' and the 'Operation Data Structure' (PDUs). To make this categorisation clearer, it would be better stated as:

- Service Common Definitions
  - Data Types
  - Operation Structures (PDUs)

- Service Specific Definitions
  - Data Types
  - Operation Structures (PDUs)

For the analysis of the operation parameters we therefore need to identify for each parameter of an operation whether it is a Service Common 'data type' or Service Specific 'data type'. This only indicates whether the type of the parameter is common across services and not necessarily that the parameter itself is common within all services – for example, the START operation has a parameter 'start-time'. This parameter would be a Service Common Data Type (some common definition of 'time') but as can be seen in Section 4.1.2.1 it is only applicable to Return Service Invocations. Therefore the 'Operation Structure' (PDU) used for the START Operation will be a Service Specific Operation Structure (which will include both Service Common Data Types and Service Specific Data Types).

If ALL parameters can be defined as being of Service Common Data Type then logically the Operation Structure will be Service Common. However, if one or more parameters are of Service Specific Data Type or a particular parameter is not applicable for ALL services, then the Operation Structure will be Service Specific.

In addition, we will also suggest potential harmonisation of parameters, where it is appropriate.

3.1.3 Association Operations

An association is a cooperative relationship between a SLE service-using application and a SLE service-providing application. The Association element consists of the BIND, UNBIND and PEER-ABORT operations. The CCSDS Recommendations for Forward Services (CLTU/FSP) show that the BIND and UNBIND operation are invoked by the User whilst for the Return Services (ROCF/RCF/RAF)

they can be invoked by User or Provider. It should be noted that the ESA SLE API only supports User invoked BIND and UNBIND.

3.1.3.1 BIND

The BIND operation is used to establish an association between a SLE service-using application and a SLE service-providing application. It is a 'confirmed operation' and the existing SLE services require the following parameters for the BIND operation:

BIND	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
initiator-identifier	M		M		M		M		M	
responder-identifier		M		M		M		M		M
responder-port-identifier	M		M		M		M		M	
service-type	M		M		M		M		M	
version-number	M	C	M	C	M	C	M	C	M	C
service-instance-identifier	M		M		M		M		M	
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

Table 3-3: Association Element – BIND Operation Parameters

The BIND operation is a 'confirmed operation' – the initiator invokes the BIND and the responder returns (or replies) on the outcome of the BIND operation.

As can be seen from the table above, the indicated level of commonality for the Association element BIND operation is high. The required invocation parameters (I) are identical for each service and all are mandatory. For the return parameters (R) the list of parameters is also identical for each service, and the conditional parameters *'version-number'* and *'diagnostic'* depend on the value of the *'result'* parameter.

In fact, all the parameters for the BIND operation are of Service Common Data Type and therefore the Operation Structure would also be Service Common.

3.1.3.2 UNBIND

The UNBIND operation is used (with the ESA SLE API, by the initiator) to release an association previously established by a BIND. Like the BIND operation the UNBIND operation is a 'confirmed operation'. The existing SLE services require the following parameters for the UNBIND operation:

UNBIND	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**

UNBIND	RAF		RCF		ROCF		CLTU		FSP	
Invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
Unbind-reason	M		M		M		M		M	
Result		M		M		M		M		M

Table 3-4: Association Element – UNBIND Operation Parameters

From the table above, the indicated level of commonality for the Association element UNBIND operation appears high. The list of required invocation (I) and return (R) parameters are identical for each service and all are mandatory.

As with the BIND operation, all the parameters are of a Service Common Data Type and therefore the UNBIND Operation Structure would also be Service Common.

3.1.3.3 PEER-ABORT

Either the SLE Service User or the SLE Service Provider can invoke the PEER-ABORT operation. It is used to notify the peer that the local SLE application detected an error that requires the association to be terminated.

It is an 'unconfirmed operation' and the existing SLE services require the following parameter for the PEER-ABORT operation:

PEER-ABORT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Diagnostic	M		M		M		M		M	

Table 3-5: Association Element – PEER-ABORT Operation Parameters

As can be seen from the table above, the required invocation parameter (I) is identical and mandatory for each service. The parameter is of Service Common Data Type and therefore the PEER-ABORT Operation Structure would also be Service Common.

3.1.4 Data Exchange Operations

Excluding the Association operations, all other SLE operations are related to data exchange/transfer between a SLE service-using application and a SLE service-providing application. This consists of the START, STOP, TRANSFER-DATA, SCHEDULE-STATUS-REPORT, STATUS-REPORT, SYNC-NOTIFY, ASYNC-NOTIFY, THROW-EVENT and GET-PARAMETER operations.

3.1.4.1 START

The START operation is invoked by the SLE Service User and is a 'confirmed operation'. Functionally it has a slightly different purpose for RETURN Services compared to FORWARD Services. For a RETURN Service it is a request for the SLE Service Provider to start the delivery of 'data' to the User

via the TRANSFER-DATA operation. In comparison, for a FORWARD Service, it is a request that the SLE Service Provider prepare to accept the delivery of 'data' from the User via the TRANSFER-DATA operation.

The existing SLE services require the following parameters for the START operation:

START	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Invoker-credentials	M		M		M		M		M	
Performer-credentials		M		M		M		M		M
Invoke-ID	M	M	M	M	M	M	M	M	M	M
start-time	M		M		M					
stop-time	M		M		M					
requested-frame-quality	M									
Global-VCID			M		M					
control-word-type					M					
tc-vcid					C					
update-mode					M					
first-cltu-identification							M			
first-packet-identification									M	
start-production-time								C		C[4]
stop-production-time								C		C[4]
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

Table 3-6: Operation Element – START Operation Parameters

The START operation is a 'confirmed operation'. The following parameters can be categorised as of a Service Common Data Type:

- invoker-credentials
- performer-credentials
- Invoke-ID
- start-time
- stop-time
- Result

[4] It is noted that this is different from that stated in [FSP-Blue], but that the CCSDS Recommendation document is incorrect and is being updated. These parameters are therefore the same as for the CLTU Service.

It should be noted however, that even though the start-time and stop-time parameters are of a Service Common Data Type that this parameter is not required across ALL services.

The remaining parameters are of a Service Specific Data Type. Due to this the START Operation Structure is also Service Specific.

The following parameters have been identified which could be harmonised:

Exisiting Parameter Name	Proposed Parameter Name
start-time	service-start-time
start-production-time	
stop-time	service-stop-time
stop-production-time	

It should be noted that the start-time and stop-time parameters are Mandatory/Invocation ones whilst the start-production-time and stop-production-time are Conditional/Return parameters.

Exisiting Parameter Name	Proposed Parameter Name
first-cltu-identification	first-dataitem-identification
first-packet-identification	

3.1.4.2 STOP

The STOP operation is invoked by the SLE Service User and is a 'confirmed operation'. For a RETURN Service it is a request to the SLE Service Provider to stop the delivery of 'data' to the User. For a FORWARD Service, it is a request to the SLE Service Provider to stop service provision and production.

The existing SLE services require the following parameters for the STOP operation:

STOP	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
invoke-ID	M	M	M	M	M	M	M	M	M	M
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

Table 3-7: Operation Element – STOP Operation Parameters

As can be seen from the table above, the indicated level of commonality for the STOP Operation appears high. The required invocation parameters (I) are identical for each service and all are

mandatory. For the return parameters (R) the list of parameters is identical for each service, and the conditional parameter *'diagnostic'* depends on the value of the *'result'* parameter.

Like the START operation the STOP operation is also a 'confirmed operation'. All the parameters are of a Service Common Data Type and therefore the STOP Operation Structure would also be Service Common.

3.1.4.3 TRANSFER-DATA

This operation is used to exchange/transfer data between a SLE service-using application and a SLE service-providing application. For RETURN Services, the operation is invoked by the Provider to transfer data to the SLE Service User and is an 'unconfirmed operation'. However, for FORWARD Services, the operation is invoked by the User to transfer data to the SLE Service Provider and is a 'confirmed operation'. The existing SLE services require the following parameters for the TRANSFER-DATA operation:

TRANSFER-DATA	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
Earth-receive-time	M		M		M					
antenna-ID	M		M		M					
data-link-continuity	M		M		M					
delivered-frame-quality	M									
private-annotation	M		M		M					
performer-credentials								M		M
invoke-ID							M	M	M	M
cltu-identification							M			
earliest-radiation-time							M			
latest-radiation-time							M			
packet-identification									M	M
earliest-production-time									M	
latest-production-time									M	
delay-time							M		M	
Report							M			
Transmission-mode									M	
MAP-identifier									M	
Blocking									M	
processing-started-notification									M	
radited-notification									M	

TRANSFER-DATA	RAF		RCF		ROCF		CLTU		FSP	
acknowledged-notification									M	
Data	M		M		M		M		M	
cltu-buffer-available								M		
packet-buffer-available										M
Result								M		M
Diagnostic								C		C

Table 3-8: Operation Element – TRANSFER-DATA Operation Parameters

The TRANSFER-DATA operation includes the following parameters that are of a Service Common Data Type:

- invoker-credentials
- Earth-receive-time[5]
- performer-credentials[5]
- Invoke-ID[5]
- earliest-radiation-time[5]
- latest-radiation-time[5]
- earliest-production-time[5]
- latest-production-time[5]
- delay-time[5]
- Result[5]

The remaining parameters are of a Service Specific Data Type. Due to this the TRANSFER-DATA Operation Structure is also Service Specific.

The following parameters have been identified which could be harmonised:

Exisiting Parameter Name	Proposed Parameter Name
cltu-identification	Dataitem-identification
packet-identification	
earliest-radiation-time	Earliest-service-time
earliest-production-time	
latest-radiation-time	Latest-service-time
latest-production-time	
cltu-buffer-available	Remaining-databuffer-available

[5] Parameter is of a Service Common Data Type but not required across ALL services

Exisiting Parameter Name	Proposed Parameter Name
packet-buffer-available	

3.1.4.4 SCHEDULE-STATUS-REPORT

This operation is invoked by the SLE Service User to request that the Provider send a status report immediately, periodically or to stop sending periodic status reports. It is a 'confirmed operation' and the existing SLE services require the following parameters for the SCHEDULE-STATUS-REPORT operation:

SCHEDULE-STATUS-REPORT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
Invoke-ID	M	M	M	M	M	M	M	M	M	M
report-request-type	M		M		M		M		M	
reporting-cycle	C		C		C		C		C	
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

Table 3-9: Operation Element – SCHEDULE-STATUS-REPORT Operation Parameters

As can be seen from the table above, the indicated level of commonality for the SCHEDULE-STATUS-REPORT Operation appears high. The required invocation parameters (I) are identical for each service and the conditional parameter *'reporting-cycle'* depends on the value of *'reporting-request-type'*. For the return parameters (R) the list of parameters is identical for each service, and the conditional parameter *'diagnostic'* depends on the value of the *'result'* parameter.

In fact, all the parameters for the SCHEDULE-STATUS-REPORT operation are of a Service Common Data Type. The *'reporting-cycle'* parameter is considered Service Common, as it is simply the cycle duration expressed in seconds. The *'diagnostic'* parameter, which for many other operations indicates Service Specific information for a negative Result, is Service Common for this particular operation. Therefore the SCHEDULE-STATUS-REPORT Operation Structure is also Service Common.

3.1.4.5 STATUS-REPORT

This operation is invoked by the SLE Service Provider to send a status report to the SLE Service User. It is an 'unconfirmed operation' and the existing SLE services require the following parameters for the STATUS-REPORT operation:

STATUS-REPORT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**

STATUS-REPORT	RAF		RCF		ROCF		CLTU		FSP	
Invoker-credentials	M		M		M		M		M	
number-of-error-free-frames-delivered	M									
number-of-frames-delivered	M		M							
number-of-frames-processed					M					
number-of-ocfs-delivered					M					
frame-sync-lock-status	M		M		M					
symbol-sync-lock-status	M		M		M					
Subcarrier-lock-status	M		M		M					
carrier-lock-status	M		M		M					
cltu-last-processed							M			
cltu-last-OK							M			
cltu-status							C			
radiation-start-time							C			
radiation-stop-time							C			
packet-identification-last-processed									M	
production-start-time									C	
packet-status									C	
packet-identification-last-OK									M	
production-stop-time									C	
production-status	M		M		M		M		M	
uplink-status							M			
number-of-cltus-received							M			
number-of-cltus-processed							M			
number-of-cltus-radiated							M			
cltu-buffer-available							M			
number-of-packets-received							M			
number-of-packets-processed							M			
number-of-packets-radiated							M			
number-of-packets-acknowledged							M			
packet-buffer-available							M			

Table 3-10: Operation Element – STATUS-REPORT Operation Parameters

As could be expected, the STATUS-REPORT Operation contains many Service Specific parameters. Those that can be considered as a Service Common Data Type though are:

- invoker-credentials

- radiation-start-time[6]
- radiation-stop-time[6]
- production-start-time[6]
- production-stop-time[6]
- production-status

It is also worth mentioning/considering a Service Common Data Type for the many 'counts' (parameters starting with *'number-of-…'*) that are required.

However, as the remaining parameters are of a Service Specific Data Type, the STATUS-REPORT Operation Structure is also Service Specific.

The following parameters have been identified which could be harmonised:

Exisiting Parameter Name	Proposed Parameter Name
number-of-frames-delivered	number-of-dataitems-delivered
number-of-ocfs-delivered	
cltu-last-processed	dataitem-last-processed
packet-identification-last-processed	
cltu-last-OK	dataitem-last-OK
packet-identification-last-OK	
radiation-start-time	service-start-time
production-start-time	
radiation-stop-time	service-stop-time
production-stop-time	
number-of-cltus-received	number-of-dataitems-received
number-of-packets-received	
number-of-cltus-processed	number-of-dataitems-processed
number-of-packets-processed	
number-of-cltus-radiated	number-of-dataitems-radiated
number-of-packets-radiated	
cltu-buffer-available	remaining-databuffer-available
packet-buffer-available	

[6] Parameter is of a Service Common Data Type but not required across ALL services

3.1.4.6 NOTIFY OPERATIONS

There are TWO notification events described in the CCSDS Recommendation documents, SYNC-NOTIFY used by RETURN Services and ASYNC-NOTIFY used by FORWARD Services. They are both invoked by the SLE Service Provider to notify the User of an event affecting production or provision of the service.

We have considered the operations and parameters separately in the sections below.

3.1.4.6.1 SYNC-NOTIFY

This operation is invoked by the SLE Service Provider to notify the User of an event affecting production or provision of the service. It is an 'unconfirmed operation' and the existing SLE services require the following parameters for the SYNC-NOTIFY operation:

SYNC-NOTIFY	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**		**I**		**I**		**I**		**I**	
invoker-credentials	M		M		M					
notification-type	M		M		M					
notification-value	C		C		C					

Table 3-11: Operation Element – SYNC-NOTIFY Operation Parameters

As can be seen from the table above, this operation is only applicable for RETURN services. The required invocation parameters (I) are identical for each return service and the conditional parameter *'notification-value'* depends on the value of *'notification-type'.*

The invoker-credentials parameter is of a Service Common Data Type. The *'notification'* parameters (both -type and -value) can be considered common, but only in the context of Return Services. We therefore have to consider the SYNC-NOTIFY Operation Structure as Service Specific.

3.1.4.6.2 ASYNC-NOTIFY

This operation is also invoked by the SLE Service Provider to notify the User of an event affecting production or provision of the service. It is an 'unconfirmed operation' and the existing SLE services require the following parameters for the ASYNC-NOTIFY operation:

ASYNC-NOTIFY	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials							M		M	
Notification-type							M		M	
directive-executed-identification									C	
event-thrown-identification							C		C	
cltu-last-processed							M			

ASYNC-NOTIFY	RAF		RCF		ROCF		CLTU		FSP	
cltu-last-OK							M			
cltu-status							M			
radiation-start-time							C			
radiation-stop-time							C			
packet-identification-list									C	
fop-alert									C	
packet-identification-last-processed									M	
Production-start-time									C	
packet-status									C	
packet-identification-last-ok									M	
Production-stop-time									C	
Production-status							M		M	
uplink-status							M			

Table 3-12: Operation Element – ASYNC-NOTIFY Operation Parameters

As can be seen from the table above, this operation is only applicable for FORWARD services. The parameters that are of a Service Common Data Type are:

- invoker-credentials
- radiation-start-time[7]
- radiation-stop-time[7]
- production-start-time[7]
- production-stop-time[7]
- production-status

However, as the remaining parameters are of a Service Specific Data Type, the ASYNC-NOTIFY Operation Structure is also Service Specific.

The following parameters have been identified which could be harmonised:

Exisiting Parameter Name	Proposed Parameter Name
cltu-last-processed	dataitem-last-processed
packet-identification-last-processed	
cltu-last-OK	dataitem-last-OK
packet-identification-last-OK	

[7] Parameter is of a Service Common Data Type but not required across ALL services

Exisiting Parameter Name	Proposed Parameter Name
cltu-status	dataitem-status
packet-status	
radiation-start-time	service-start-time
production-start-time	
radiation-stop-time	service-stop-time
production-stop-time	

It should be noted that most of the above parameter names are identical to those for the STATUS-REPORT operation.

3.1.4.7 THROW-EVENT

A SLE Service User invokes the THROW-EVENT operation in order to cause the provider to forward to SLE Complex Management an event that requires management action. It is a 'confirmed operation' that is currently only used by FORWARD Services.

The existing SLE services require the following parameters for the THROW-EVENT operation:

THROW-EVENT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials							M		M	
performer-credentials								M		M
invoke-ID							M	M	M	M
event-invocation-identification							M	M	M	M
event-identifier							M		M	
event-qualifier							M		M	
Result								M		M
Diagnostic								C		C

Table 3-13: Operation Element – THROW-EVENT Operation Parameters

As can be seen from the table above, this operation is only applicable for FORWARD services. The required invocation parameters (I) are identical for each service and all are mandatory. For the return parameters (R) the list of parameters is identical for each service, and the conditional parameter *'diagnostic'* depends on the value of the *'result'* parameter.

The parameters that are of a Service Common Data Type (though none are obviously applicable to ALL services) are:

- invoker-credentials
- performer-credentials

- invoke-ID
- result
- diagnostic

Of the other parameters, *'event-invocation-identification'* is simply a monotonically increasing sequence number, *'event-identifier'* is represented as an Integer identifying the event to be forwarded to Complex Management and the '*event-qualifier'* parameter is a fixed-length octet string used to provide additional data applicable to the event. These parameters could therefore ALSO be considered as of a Service Common Data Type.

Strictly speaking, as the THROW-EVENT operation is currently only used by Forward Services, it would mean that the Operation Structure is Service Specific. However, it is certainly conceivable that for certain operational scenarios it could be desirable to have the THROW-EVENT operation for Return Services as well. One such scenario could be a spacecraft that autonomously changes telemetry data-rate in response to a particular onboard event.

Therefore, as all parameters have been identified as being of a Service Common Data Type, it would seem sensible to consider the THROW-EVENT Operation Structure as Service Common.

3.1.4.8 GET-PARAMETER

The GET-PARAMETER operation is invoked by a SLE Service User to ascertain the value of an SLE Service parameter. It is a 'confirmed operation' and the existing SLE services require the following parameters for the operation:

GET-PARAMETER	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
invoke-ID	M	M	M	M	M	M	M	M	M	M
raf-parameter	M	C								
rcf-parameter			M	C						
rocf-parameters					M	C				
cltu-parameter							M	C		
fsp-parameter									M	C
parameter-value		C		C		C		C		C
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

Table 3-14: Operation Element – GET-PARAMETER Operation Parameters

It can be seen from the above table, that the required invocation parameters (I) are identical except for the service specific *'parameter-data'* (to be expected). For the return parameters (R) it is the same, and the conditional parameters depend on the value of the *'result'* parameter.

The parameters that are of a Service Common Data Type are:

- invoker-credentials
- performer-credentials
- invoke-ID
- Result
- Diagnostic

In the current SLE API implementation the *'ServiceType-parameter'* parameter, is specified in the Service Common definition as a 'named-integer'. The *'parameter-value'* parameter is specified at the Service Specific level, where the data-type for the value is dependent on the parameter name. It is therefore conceivable that the GET-PARAMETER Operation Structure can be considered as Service Common even though (actual) implementation details are specified at a Service Specific level for the data-type of the *'ServiceType-parameter' and 'parameter-value'* parameters.

The following parameters have been identified which could be harmonised:

Exisiting Parameter Name	Proposed Parameter Name
raf-parameter	service-parameter-name
rcf-parameter	
rocf-parameters	
cltu-parameter	
fsp-parameter	

With the above change, it would also be more consistent to change the name of the *'parameter-value'* parameter to *'service-parameter-value'*.

3.1.4.9 INVOKE-DIRECTIVE

The INVOKE-DIRECTIVE operation is specific to the FSP Service. This operation is invoked by a SLE Service User to invoke TC Directives required to (re-)establish commanding capability on a given VC. It is a 'confirmed operation' and requires the following parameters:

INVOKE-DIRECTIVE	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials									M	
performer-credentials										M
invoke-ID									M	M
directive-identification									M	M

INVOKE-DIRECTIVE	RAF		RCF		ROCF		CLTU		FSP	
directive									M	
Result										M
Diagnostic										C

Table 3-15: Operation Element – INVOKE-DIRECTIVE Operation Parameters

The parameters that are of a Service Common Data Type are:

- invoker-credentials
- performer-credentials
- invoke-ID
- Result

The *'directive-identification'* parameter is a monotonically increasing sequence number that has been implemented as an Integer. The *'directive'* parameter consists of a directive-code and, where applicable, an associated parameter. Similar to the *'ServiceType-parameter'* parameter of the GET-PARAMETER, it has been implemented as a 'named-integer' but at the Service Specific level.

Even though the INVOKE-DIRECTIVE Operation is only applicable to the FSP Service, we could therefore consider the Operation Structure Service Common.

3.1.5 Summary

To summarise the commonality highlighted in the Operations and Parameters we can say that:

1. There are a number of Service Common Data Types and these can be categorised as:

   1.1. Parameters related to 'authentication'
   Eg: invoker-credentials, performer-credentials, initiator-identifier and responder-identifier

   1.2. Parameters which are 'identifiers'
   Eg: service-instance-identifier, responder-port-identifier

   1.3. Time-based parameters
   Eg: start-time, stop-time

   1.4. Number-based parameters (including 'counters')
   Eg: version-number, Result

2. There are a few Service Common Operation Structures where ALL the parameters are of a Service Common Data Type. These are:

   2.1. Association Operations
   2.1.1. BIND
   2.1.2. UNBIND
   2.1.3. PEER-ABORT

   2.2. Data Exchange Operations
   2.2.1. STOP
   2.2.2. SCHEDULE-STATUS-REPORT

3. There are a few Service Common Operation Structures where SOME of the parameters are of a Service Specific Data Type. These are:

   3.1. Data Exchange Operations
   3.1.1. THROW-EVENT
   3.1.2. GET-PARAMETER
   3.1.3. INVOKE-DIRECTIVE

4. The following Operations have Service Specific Operation Structures including Service Specific Data Types

   4.1. Data Exchange Operations
   4.1.1. START
   4.1.2. TRANSFER-DATA
   4.1.3. STATUS-REPORT
   4.1.4. SYNC-NOTIFY
   4.1.5. ASYNC-NOTIFY

On the harmonisation of the parameter names we have identified, for some of the Data Exchange operations, a number of parameters for which the names could be rationalised. These seem to fall into one of two categories:

- Related to time

  These include start/stop as well as earliest/latest time parameters. Care needs to be taken when considering the use of processing/radiation/production times (mainly used in the Forward Services) as these could have subtle differences.

- Including ‘cltu’ or ‘packet’ in the parameter name for the Forward Services

  The Forward Services seem to include the ‘data item’ (cltu/packet) in many of the parameter names. Whilst this does give some clarity, it does not allow support/extension for new Forward Services. We have suggested replacing the use of cltu/packet with ‘dataitem’ as a generalisation.

It must be noted though that these modifications would have an impact on backward compatibility that needs to be considered.

3.2 Service Provider State Transitions

The CCSDS Recommendation documents, [CLTU-Blue], [FSP-Blue], [RAF-Blue], [RCF-Blue] and [ROCF-Blue], consider the State Transitions for a Service Provider. A simplified state transition diagram is shown below:

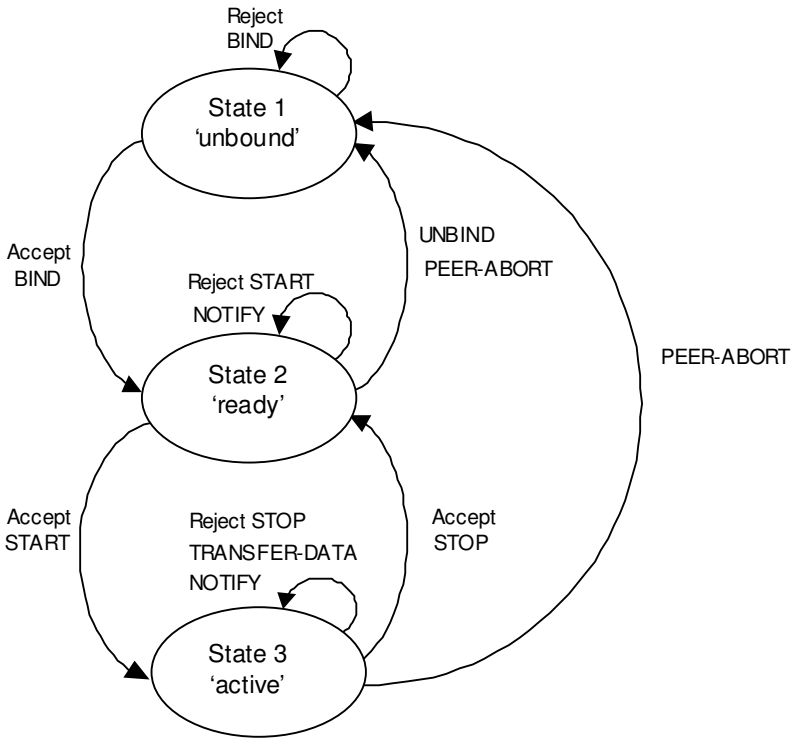

Figure 3-1: Simplified Service Provider State Transition Diagram

Where:

1. State 1 ('unbound')

   All resources required to enable the provision of the 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 transfer service provider port is not bound).

2. State 2 ('ready')

   An association has been established between the user and the provider. However, the transfer of data between user and provider (by means of the TRANSFER-DATA operation) is not permitted. The user may enable this by means of the appropriate service START operation, which, in turn, will cause the provider to transition to state 3 ('active').

3. State 3 ('active')

State 3 resembles state 2 (‘ready’), except that now the transfer of data between user and provider can occur. The service continues in this state until the user invokes the STOP operation to cause the provider to transition back to state 2.

For the commonality analysis we have reviewed the State Transition Matrices from Section 4 of the CCSDS Recommendation documents (the individual Service State Transition Matrix are included in the Appendices of this document). We have considered the Forward Services separate from the Return Services and compared the behaviour performed upon reception of an Incoming Event.

The following sections contain the comparison of the state tables of the forward and return services. The events within the state tables of the services have been grouped into similar ‘event types’, i.e. for example a bind invocation, and for each group the state transition is shown. This grouping allows a simple comparison of the similarities of the state transitions, depending on event type.

3.2.1 Commonality of Events For Forward Service Providers

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FGEN-1	(genBindInvocation)	IF "positive result" THEN (+genBindReturn) → 2 ELSE (-genBindReturn)	IF "same service instance" THEN (-genBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1	IF "same service instance" THEN (-genBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1
CLTU-1	(cltuBindInvocation)	IF "positive result" THEN (+cltuBindReturn) → 2 ELSE (-cltuBindReturn)	IF "same service instance" THEN (-cltuBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1	IF "same service instance" THEN (-cltuBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1
FSP-1	(fspBindInvocation)	IF "positive result" .AND. "production configured" THEN {accept bind} → 2 ELSE IF "positive result" THEN (+fspBindReturn) → 2 ELSE (-fspBindReturn)	IF "same service instance" THEN (-fspBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1	IF "same service instance" THEN (-fspBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1
FGEN-2	'end of service instance provision period'	[ignore]	{peer abort ('end of service instance provision period')} → 1	{peer abort ('end of service instance provision period')} → 1
CLTU-2	'end of service instance provision period'	[ignore]	{peer abort ('end of service instance provision period')} → 1	{peer abort ('end of service instance provision period')} → 1
FSP-2	'end of service instance provision period'	[ignore]	{peer abort ('end of service instance provision period')} → 1	{peer abort ('end of service instance provision period')} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FGEN-3	(genUnbindInvocation)	[ignore]	(genUnbindReturn) → 1 stop reporting-cycle timer IF "end" THEN release resources ELSE [ignore]	{peer abort ('protocol error')} → 1
CLTU-3	(cltuUnbindInvocation)	[ignore]	(cltuUnbindReturn) → 1 stop reporting-cycle timer IF "end" THEN release resources ELSE [ignore]	{peer abort ('protocol error')} → 1
FSP-3	(fspUnbindInvocation)	[ignore]	(fspUnbindReturn) → 1 stop reporting-cycle timer stop all return-timeout-period timers set "notify production operational" to FALSE IF "end" THEN release resources ELSE [ignore]	{peer abort ('protocol error')} → 1
FGEN-4	(genStartInvocation)	[ignore]	IF "positive result" THEN (+genStartReturn) → 3 ELSE (-genStartReturn)	{peer abort ('protocol error')} → 1
CLTU-4	(cltuStartInvocation)	[ignore]	IF "positive result" THEN (+cltuStartReturn) → 3 ELSE (-cltuStartReturn)	{peer abort ('protocol error')} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FSP-4	(fspStartInvocation)	[ignore]	IF "positive result" THEN (+fspStartReturn) → 3 ELSE IF "production off" THEN {reject start} ELSE (-fspStartReturn)	{peer abort ('protocol error')} → 1
FGEN-5	(genStopInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" THEN {initiate stop} → 2 ELSE (-genStopReturn)
CLTU-5	(cltuStopInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" THEN {initiate stop} → 2 ELSE (-cltuStopReturn)
FSP-5	(fspStopInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" THEN {initiate stop} → 2 ELSE (-fspStopReturn)
FGEN-6	(genTransferDataInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") THEN queue data (+genTransferDataReturn) ELSE discard data (-genTransferDataReturn)

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
CLTU-6	(cltuTransferDataInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") THEN buffer CLTU (+cltuTransferDataReturn) ELSE discard CLTU (-cltuTransferDataReturn)
FSP-6	(fspTransferDataInvocation) with BD Packet	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") THEN queue packet (+fspTransferDataReturn) ELSE discard packet (-fspTransferDataReturn)
FSP-7	(fspTransferDataInvocation) with AD Packet and unblock AD	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") .AND. ("AD blocked") THEN queue packet Set "AD blocked" to FALSE (+fspTransferDataReturn) ELSE discard packet (-fspTransferDataReturn)
FSP-8	(fspTransferDataInvocation) with AD Packet and .NOT. unblock AD	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") .AND. (.NOT. "AD blocked") THEN queue packet (+fspTransferDataReturn) ELSE discard packet (-fspTransferDataReturn)

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FGEN-7	(genScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+genScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-genScheduleStatusReportReturn)	IF "positive result" THEN (+genScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-genScheduleStatusReportReturn)
CLTU-7	(cltuScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+cltuScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-cltuScheduleStatusReportReturn)	IF "positive result" THEN (+cltuScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-cltuScheduleStatusReportReturn)
FSP-9	(fspScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+fspScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-fspScheduleStatusReportReturn)	IF "positive result" THEN (+fspScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-fspScheduleStatusReportReturn)
FGEN-8	'reporting-cycle timer expired'	Not applicable	{periodic report}	{periodic report}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
CLTU-8	'reporting-cycle timer expired'	Not applicable	{periodic report}	{periodic report}
FSP-10	'reporting-cycle timer expired'	Not applicable	{periodic report}	{periodic report}
FGEN-9	'return-timeout-period timer <n> expired'	Not applicable	{peer abort ('return timeout')} → 1	{peer abort ('return timeout')} → 1
CLTU-9	'return-timeout-period timer <n> expired'	Not applicable	{peer abort ('return timeout')} → 1	{peer abort ('return timeout')} → 1
FSP-11	'return-timeout-period timer <n> expired'	Not applicable	{peer abort ('return timeout')} → 1	{peer abort ('return timeout')} → 1
FGEN-10	(genGetParameterInvocation)	[ignore]	IF "positive result" THEN (+genGetParameterReturn) ELSE (-genGetParameterReturn)	IF "positive result" THEN (+genGetParameterReturn) ELSE (-genGetParameterReturn)
CLTU-10	(cltuGetParameterInvocation)	[ignore]	IF "positive result" THEN (+cltuGetParameterReturn) ELSE (-cltuGetParameterReturn)	IF "positive result" THEN (+cltuGetParameterReturn) ELSE (-cltuGetParameterReturn)
FSP-12	(fspGetParameterInvocation)	[ignore]	IF "positive result" THEN (+fspGetParameterReturn) ELSE (-fspGetParameterReturn)	IF "positive result" THEN (+fspGetParameterReturn) ELSE (-fspGetParameterReturn)
FGEN-11	(genThrowEventInvocation)	[ignore]	IF "positive result" THEN (+genThrowEventReturn) forward event to Complex Management ELSE (-genThrowEventReturn)	IF "positive result" THEN (+genThrowEventReturn) forward event to Complex Management ELSE (-genThrowEventReturn)
CLTU-11	(cltuThrowEventInvocation)	[ignore]	IF "positive result" THEN (+cltuThrowEventReturn) forward event to Complex Management ELSE (-cltuThrowEventReturn)	IF "positive result" THEN (+cltuThrowEventReturn) forward event to Complex Management ELSE (-cltuThrowEventReturn)

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FSP-13	(fspThrowEventInvocation)	[ignore]	IF "positive result" THEN (+fspThrowEventReturn) forward event to Complex Management ELSE (-fspThrowEventReturn)	IF "positive result" THEN (+fspThrowEventReturn) forward event to Complex Management ELSE (-fspThrowEventReturn)
FSP-14	(fspInvokeDirectiveInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" THEN queue directive (+fspInvokeDirectiveReturn) ELSE (-fspInvokeDirectiveReturn)
FSP-15	'packet processing started'	Not applicable	Not applicable	IF "report processing" THEN {notify ('packet processing started')} ELSE [ignore]
FSP-16	'packet acknowledged'	Not applicable	IF "report acknowledgement" THEN {notify ('packet acknowledged')} ELSE [ignore]	IF "report acknowledgement" THEN {notify ('packet acknowledged')} ELSE [ignore]
FGEN-12	'data radiated'	[ignore]	IF "report radiation" THEN {notify} ELSE [ignore]	IF "report radiation" THEN {notify} ELSE [ignore]
CLTU-12	'cltu radiated'	[ignore]	IF "report" THEN {notify('cltu radiated')} ELSE [ignore]	IF "report" THEN {notify('cltu radiated')} ELSE [ignore]
FSP-17	'packet radiated'	Not applicable	IF "report radiation" THEN {notify ('packet radiated')} ELSE [ignore]	IF "report radiation" THEN {notify ('packet radiated')} ELSE [ignore]

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FGEN-13	'sldu expired'	*Service Specific*	Not applicable	{notify('sldu expired') and block}
CLTU-13	'sldu expired'	IF "continue" THEN clear CLTU buffer ELSE [ignore]	Not applicable	{notify('sldu expired') and block}
FSP-18	'sldu expired'	Not applicable	Not applicable	{notify ('sldu expired') and block}
FSP-19	'packet transmission mode mismatch'	Not applicable	Not applicable	{notify ('packet transmission mode mismatch') and block AD}
FSP-20	'transmission mode capability change'	Not applicable	{notify ('transmission mode capability change')}	{notify ('transmission mode capability change')}
FSP-21	'VC aborted'	Not applicable	{notify ('VC aborted')}	{notify ('VC aborted')}
FGEN-14	'production interrupted'	*Service Specific*	*Service Specific*	{notify('production interrupted') and block} set "notify production operational" to TRUE
CLTU-14	'production interrupted'	IF "continue" THEN clear CLTU buffer ELSE [ignore]	{notify('production interrupted') and clear}	{notify('production interrupted') and block} set "notify production operational" to TRUE
FSP-22	'production interrupted'	Not applicable	Not applicable	{notify ('production interrupted') and block} set "notify production operational" to TRUE
FGEN-15	'production halted'	*Service Specific*	*Service Specific* Dependent on 'clear' for CLTU and whether this would also be applicable for FSP	{notify('production halted') and block} set "notify production operational" to TRUE
CLTU-15	'production halted'	IF "continue" THEN clear CLTU buffer ELSE [ignore]	{notify('production halted') and clear} set "notify production operational" to TRUE	{notify('production halted') and block} set "notify production operational" to TRUE

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FSP-23	'production halted'	Not applicable	{notify ('production halted')} set "notify production operational" to TRUE	{notify ('production halted') and block} set "notify production operational" to TRUE
FGEN-16	'production operational'	[ignore]	IF "notify production operational" THEN {notify('production operational')} set "notify production operational" to FALSE	IF "notify production operational" THEN {notify('production operational')} set "notify production operational" to FALSE
CLTU-16	'production operational'	[ignore]	IF "notify production operational" THEN {notify('production operational')} set "notify production operational" to FALSE	IF "notify production operational" THEN {notify('production operational')} set "notify production operational" to FALSE
FSP-24	'production operational'	Not applicable	IF "notify production operational" THEN {notify ('production operational')} set "notify production operational" to FALSE	IF "notify production operational" THEN {notify ('production operational')} set "notify production operational" to FALSE
FGEN-17	'buffer empty'	[ignore]	*Service Specific*	{notify('buffer empty')}
CLTU-17	'buffer empty'	[ignore]	Not applicable	{notify('buffer empty')}
FSP-25	'buffer empty'	Not applicable	{notify ('buffer empty')}	{notify ('buffer empty')}
FSP-26	'no invoke directive capability on this VC'	Not applicable	{notify ('no invoke directive capability on this VC')}	{notify ('no invoke directive capability on this VC')}
FSP-27	'invoke directive capability on this VC established'	Not applicable	{notify ('invoke directive capability on this VC established')}	{notify ('invoke directive capability on this VC established')}
FSP-28	'positive confirm response to directive'	Not applicable	{notify ('positive confirm response to directive')}	{notify ('positive confirm response to directive')}
FSP-29	'negative confirm response to directive'	Not applicable	{notify ('negative confirm response to directive')}	{notify ('negative confirm response to directive')}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FGEN-18	'action list completed'	Not applicable	{notify('action list completed')}	{notify('action list completed')}
CLTU-18	'action list completed'	Not applicable	{notify('action list completed')}	{notify('action list completed')}
FSP-30	'action list completed'	Not applicable	{notify ('action list completed')}	{notify ('action list completed')}
FGEN-19	'action list not completed'	Not applicable	{notify('action list not completed')}	{notify('action list not completed')}
CLTU-19	'action list not completed'	Not applicable	{notify('action list not completed')}	{notify('action list not completed')}
FSP-31	'action list not completed'	Not applicable	{notify ('action list not completed')}	{notify ('action list not completed')}
FGEN-20	'event condition evaluated to false'	Not applicable	{notify('event condition evaluated to false')}	{notify('event condition evaluated to false')}
CLTU-20	'event condition evaluated to false'	Not applicable	{notify('event condition evaluated to false')}	{notify('event condition evaluated to false')}
FSP-32	'event condition evaluated to false'	Not applicable	{notify ('event condition evaluated to false')}	{notify ('event condition evaluated to false')}
FGEN-21	(genPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
CLTU-21	(cltuPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
FSP-33	(fspPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
FGEN-22	'protocol abort'	[ignore]	{clean up} → 1	*Service Specific*
CLTU-22	'protocol abort'	[ignore]	{clean up} → 1	IF "continue" THEN stop reporting-cycle timer → 1 ELSE {clean up} → 1
FSP-34	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
FGEN-23	'unsolicited invoke-ID'	[ignore]	{peer abort ('unsolicited invoke-ID')} → 1	{peer abort ('unsolicited invoke-ID')} → 1
CLTU-23	'unsolicited invoke-ID'	[ignore]	{peer abort ('unsolicited invoke-ID')} → 1	{peer abort ('unsolicited invoke-ID')} → 1
FSP-35	'unsolicited invoke-ID'	[ignore]	{peer abort ('unsolicited invoke-ID')} → 1	{peer abort ('unsolicited invoke-ID')} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
FGEN-24	'invalid SLE-PDU'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
CLTU-24	'invalid SLE-PDU'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
FSP-36	'invalid SLE-PDU'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
FGEN-25	'unexpected SLE-PDU'	[ignore]	{peer abort ('protocol error')} → 1	{peer abort ('protocol error')} → 1
CLTU-25	'unexpected SLE-PDU'	[ignore]	{peer abort ('protocol error')} → 1	{peer abort ('protocol error')} → 1
FSP-37	'unexpected SLE-PDU'	[ignore]	{peer abort ('protocol error')} → 1	{peer abort ('protocol error')} → 1
FGEN-26	'not authenticated SLE-PDU'	[ignore]	[ignore]	[ignore]
CLTU-26	'not authenticated SLE-PDU'	[ignore]	[ignore]	[ignore]
FSP-38	'not authenticated SLE-PDU'	[ignore]	[ignore]	[ignore]

3.2.2 Commonality of Events For Return Service Providers

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
RAF-1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
RCF-1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
ROCF-1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
RGEN-2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RAF-2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1
RCF-2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1
ROCF-2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1
RGEN-3	(-genBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RAF-3	(-rafBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RCF-3	(-rcfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
ROCF-3	(-rocfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-4	(+genBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE [ignore] ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RAF-4	(+rafBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE [ignore] ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RCF-4	(+rcfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE [ignore] ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
ROCF-4	(+rocfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE [ignore] ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RGEN-5	(genBindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+genBindReturn) → 2 ELSE (-genBindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RAF-5	(rafBindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+rafBindReturn) → 2 ELSE (-rafBindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RCF-5	(rcfBindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+rcfBindReturn) → 2 ELSE (-rcfBindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
ROCF-5	(rocfBindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+rocfBindReturn) → 2 ELSE (-rocfBindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RGEN-6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period'} → 1
RAF-6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period'} → 1
RCF-6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period'} → 1
ROCF-6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-7	(genUnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RAF-7	(rafUnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
RCF-7	(rcfUnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
ROCF-7	(rocfUnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-8	(genUnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1
RAF-8	(rafUnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1
RCF-8	(rcfUnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1
ROCF-8	(rocfUnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-9	(genStartInvocation)	[ignore]	IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+genStartReturn) → 3 initialize transfer buffer ELSE (-genStartReturn) → 2	{peer abort 'protocol error'} → 1
RAF-9	(rafStartInvocation)	[ignore]	IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+rafStartReturn) → 3 initialize transfer buffer ELSE (-rafStartReturn) → 2	{peer abort 'protocol error'} → 1
RCF-9	(rcfStartInvocation)	[ignore]	IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+rcfStartReturn) → 3 initialize transfer buffer ELSE (-rcfStartReturn) → 2	{peer abort 'protocol error'} → 1
ROCF-9	(rocfStartInvocation)	[ignore]	IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+rocfStartReturn) → 3 initialize transfer buffer ELSE (-rocfStartReturn) → 2	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-10	(genStopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {transmit buffer} (+genStopReturn) ELSE (+genStopReturn) ELSE (-genStopReturn) → 3
RAF-10	(rafStopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {transmit buffer} (+rafStopReturn) ELSE (+rafStopReturn) ELSE (-rafStopReturn) → 3
RCF-10	(rcfStopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {transmit buffer} (+rcfStopReturn) ELSE (+rcfStopReturn) ELSE (-rcfStopReturn) → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
ROCF-10	(rocfStopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF NOT "buffer empty" THEN {transmit buffer} (+rocfStopReturn) ELSE (+rocfStopReturn) ELSE (-rocfStopReturn) → 3
RGEN-11	(genStopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {pass buffer contents} (+genStopReturn) ELSE (+genStopReturn) ELSE (-genStopReturn) → 3
RAF-11	(rafStopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {pass buffer contents} (+rafStopReturn) ELSE (+rafStopReturn) ELSE (-rafStopReturn) → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RCF-11	(rcfStopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {pass buffer contents} (+rcfStopReturn) ELSE (+rcfStopReturn) ELSE (-rcfStopReturn) → 3
ROCF-11	(rocfStopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF NOT "buffer empty" THEN {pass buffer contents} (+rocfStopReturn) ELSE (+rocfStopReturn) ELSE (-rocfStopReturn) → 3
RGEN-12	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} ELSE {insert annotated frame} → 3
RAF-12	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} ELSE {insert annotated frame} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RCF-12	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} ELSE {insert annotated frame} → 3
ROCF-12	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} ELSE {insert annotated frame} → 3
RGEN-13	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated data} {start release timer} ELSE IF "buffer empty" THEN {insert annotated data} → 3 {start release timer} ELSE {insert annotated data} → 3
RAF-13	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame} → 3 {start release timer} ELSE {insert annotated frame} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RCF-13	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame} → 3 {start release timer} ELSE {insert annotated frame} → 3
ROCF-13	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated OCF} {start release timer} ELSE IF "buffer empty" THEN {insert annotated OCF} → 3 {start release timer} ELSE {insert annotated OCF} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-14	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {insert annotated data} {start release timer} ELSE {insert annotated data} {start release timer} ELSE IF "buffer empty" THEN {insert annotated data} → 3 {start release timer} ELSE {insert annotated data} → 3
RAF-14	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {insert annotated frame} {start release timer} ELSE {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame} → 3 {start release timer} ELSE {insert annotated frame} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RCF-14	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {insert annotated frame} {start release timer} ELSE {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame} → 3 {start release timer} ELSE {insert annotated frame} → 3
ROCF-14	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {insert annotated OCF} {start release timer} ELSE {insert annotated OCF} {start release timer} ELSE IF "buffer empty" THEN {insert annotated OCF} → 3 {start release timer} ELSE {insert annotated OCF} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-15	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN {increment buffer size by one} {sync notify 'data discarded'} {start release timer} ELSE [ignore]
RAF-15	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN {increment buffer size by one} {sync notify 'data discarded'} {start release timer} ELSE [ignore]
RCF-15	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN {increment buffer size by one} {sync notify 'data discarded'} {start release timer} ELSE [ignore]
ROCF-15	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {start release timer} ELSE [ignore]

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-16	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
RAF-16	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
RCF-16	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
ROCF-16	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
RGEN-17	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE → 3 {sync notify 'end of data'} {transmit buffer}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RAF-17	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE → 3 {sync notify 'end of data'} {transmit buffer}
RCF-17	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE → 3 {sync notify 'end of data'} {transmit buffer}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
ROCF-17	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}
RGEN-18	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}
RAF-18	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RCF-18	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}
ROCF-18	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}
RGEN-19	'loss of frame synchronization', "timely online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {start release timer} {sync notify 'loss of frame sync'} ELSE {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RAF-19	'loss of frame synchronization', "timely online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {start release timer} {sync notify 'loss of frame sync'} ELSE {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
RCF-19	'loss of frame synchronization', "timely online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {start release timer} {sync notify 'loss of frame sync'} ELSE {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
ROCF-19	'loss of frame synchronization', "timely online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {start release timer} {sync notify 'loss of frame sync'} ELSE {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
RGEN-20	'loss of frame synchronization', "complete online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RAF-20	'loss of frame synchronization', "complete online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
RCF-20	'loss of frame synchronization', "complete online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
ROCF-20	'loss of frame synchronization', "complete online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-21	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3
RAF-21	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3
RCF-21	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3
ROCF-21	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3
RGEN-22	(genScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+genScheduleStatusReport) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 2 (-genScheduleStatusReportReturn)	IF "positive result" THEN (+genScheduleStatusReport) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 3 (-genScheduleStatusReportReturn)

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RAF-22	(rafScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+rafScheduleStatusReport) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 2 (-rafScheduleStatusReportReturn)	IF "positive result" THEN (+rafScheduleStatusReport) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 3 (-rafScheduleStatusReportReturn)
RCF-22	(rcfScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+rcfScheduleStatusReport) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 2 (-rcfScheduleStatusReportReturn)	IF "positive result" THEN (+rcfScheduleStatusReport) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 3 (-rcfScheduleStatusReportReturn)

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
ROCF-22	(rocfScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+rocfScheduleStatusReport) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 2 (-rocfScheduleStatusReportReturn)	IF "positive result" THEN (+rocfScheduleStatusReport) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 3 (-rocfScheduleStatusReportReturn)
RGEN-23	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
RAF-23	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
RCF-23	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
ROCF-23	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
RGEN-24	(genGetParameterInvocation)	[ignore]	IF "positive result" THEN (+genGetParameterReturn) → 2 ELSE (-genGetParameterReturn) → 2	IF "positive result" THEN (+genGetParameterReturn) → 3 ELSE (-genGetParameterReturn) → 3
RAF-24	(rafGetParameterInvocation)	[ignore]	IF "positive result" THEN (+rafGetParameterReturn) → 2 ELSE (-rafGetParameterReturn) → 2	IF "positive result" THEN (+rafGetParameterReturn) → 3 ELSE (-rafGetParameterReturn) → 3
RCF-24	(rcfGetParameterInvocation)	[ignore]	IF "positive result" THEN (+rcfGetParameterReturn) → 2 ELSE (-rcfGetParameterReturn) → 2	IF "positive result" THEN (+rcfGetParameterReturn) → 3 ELSE (-rcfGetParameterReturn) → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
ROCF-24	(rocfGetParameterInvocation)	[ignore]	IF "positive result" THEN (+rocfGetParameterReturn) → 2 ELSE (-rocfGetParameterReturn)→ 2	IF "positive result" THEN (+rocfGetParameterReturn) → 3 ELSE (-rocfGetParameterReturn)→ 3
RGEN-25	(genPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
RAF-25	(rafPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
RCF-25	(rcfPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
ROCF-25	(rocfPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
RGEN-26	'invalid protocol data unit'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
RAF-26	'invalid protocol data unit'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
RCF-26	'invalid SLE-PDU'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
ROCF-26	'invalid protocol data unit'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
RGEN-27	'return SLE-PDU with unsolicited Invoke-ID'	[ignore]	{peer abort ('unsolicited Invoke-ID')} → 1	{peer abort ('unsolicited Invoke-ID')} → 1
RAF-27	'return SLE-PDU with unsolicited Invoke-ID'	[ignore]	{peer abort ('unsolicited Invoke-ID')} → 1	{peer abort ('unsolicited Invoke-ID')} → 1
RCF-27	'return SLE-PDU with unsolicited Invoke-ID'	[ignore]	{peer abort ('unsolicited Invoke-ID')} → 1	{peer abort ('unsolicited Invoke-ID')} → 1
ROCF-27	'return SLE-PDU with unsolicited Invoke-ID'	[ignore]	{peer abort ('unsolicited Invoke-ID')} → 1	{peer abort ('unsolicited Invoke-ID')} → 1
RGEN-28	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
RAF-28	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
RCF-28	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
ROCF-28	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
RGEN-29	'not authenticated SLE-PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3
RAF-29	'not authenticated SLE-PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3
RCF-29	'not authenticated SLE-PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3
ROCF-29	'not authenticated SLE-PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3

3.2.3 Summary

It must be remembered that the commonality analysis has only been performed on the Service Provider State Transition Matrices from Section 4 of the individual CCSDS Recommendation documents.

3.2.3.1 Forward Service Provider

Commonality across the Forward Services can be identified in many of the Incoming Events. However, there are a number of transition events where certain Service Specific behaviour occurs. These can be categorised as:

1. The required behaviour in certain states, for a given Incoming Event is slightly different. Here we can identify a limited subset of generic behaviour.
   - For the BindInvocation (Event No. 1), the behaviour in the Unbound State of the FSP Service is also dependent on the “production status”. This dependency also occurs for the StartInvocation (Event No. 4) in the Ready State.
   - For the UnbindInvocation (Event No. 3), there are two additional steps required for the Ready State.
   - The TransferDataInvocation (Event No. 7) – behaviour for FSP depends on the AD/BD configuration.

2. The required behaviour for some events is clearly Service Specific
   - The ‘sldu expired’ (Event No. 13) has service specific behaviour in the Unbound State.
   - The ‘production interrupted’ (Event No. 14) and ‘production halted’ *Event No. 15) have service specific behaviour in the Unbound and Ready States
   - The ‘buffer empty’ (Event No 17) has service specific behaviour in the Ready State
   - The ‘protocol abort’ (Event No. 22) has service specific behaviour in the Active State

In addition, the FSP Service has a number of Incoming Events which are totally Service Specific and there is no equivalent for the CLTU Service:

- Event FSP-19 – ‘packet transmission mode mismatch’
- Event FSP-20 – ‘transmission mode capability change’
- Event FSP-21 – VC aborted’
- Events related to the FSP-INVOKE-DIRECTIVE operation, FSP-26 – FSP-29.

3.2.3.2 Return Service Provider

Commonality across the Return Services is complete. For each Incoming Event, irrespective of the Service state, the behaviour is identical (no Service Specific behaviour required).

4 Commonality Concepts

In this chapter are a series of commonality concepts presented and discussed, all based on the 'commonality analysis' in chapter 3. The intension of the commonality concepts is to form the basis of an approach for the SLE Toolkit definition.

4.1 Operations and Parameters

One of the goals of the SLE toolkit is to promote a concept that will support each operation of all current Services, as well as encapsulating commonality in order to simplify the definition and implementation of future SLE Services.

A possible Generic Concept is indicated by the GET-PARAMETER operation, where parameters that are of a Service Common Data Type AND required for ALL service types, can be considered differently from the other parameters that are Operation/Service Specific. It can be described as a 'grouping' within the Service Common Operation Structure, as shown below:

Structure	Description	Example Parameters
HEADER	Parameters that are of a Service Common Data Type and required for ALL service types.	invoker-credentials
		performer-credentials
		invoke-ID
		Result (if present)
		Diagnostic (if present)
DATA	Indicates the type of service (could however be a Service Common Data Type)	Service-type[8] RAF/RCF/ROCF – CLTU/FSP
	This will be the most flexible area – will be a set of parameters dependent on the operation/service-type. Can contain parameters of both Service Common Data Types (but which are not required by ALL services) and Service Specific Data Types.	Service-dependent-data-parameter

Table 4-1: Generic Service Common Operation Structure

One of the ideas behind this being that the Service Common Data Types, Service Common Operation Structure and the HEADER can form part of the SLE Toolkit and the Service-dependent information is separate.

This can represented in a layered hierarchical representation as shown in the following figure:

[8] Could be argued that the 'service-type' parameter should be part of the Header as it is a Service Common Data Type

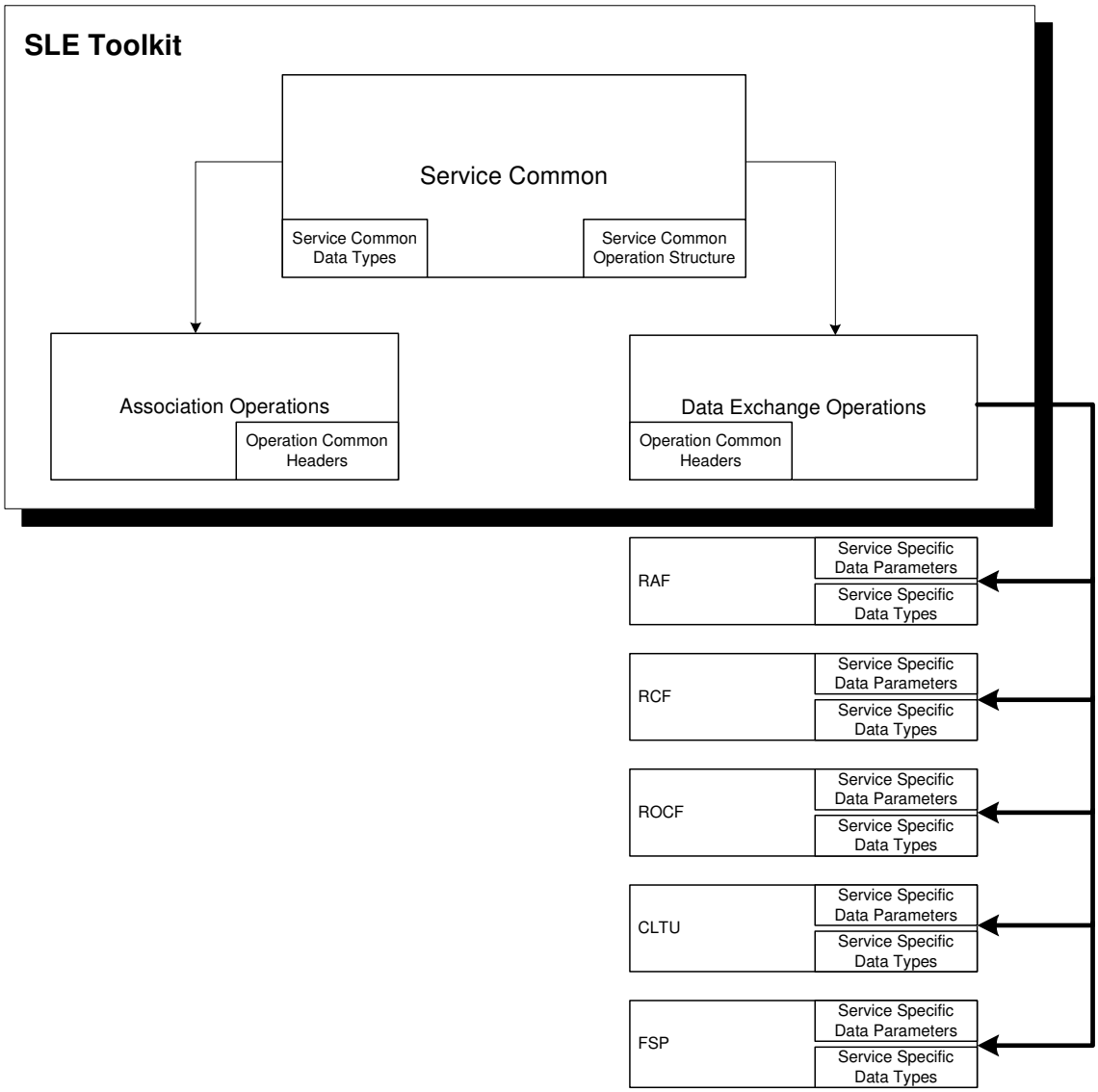

Figure 4-1: Operation and Parameter Architecture

Where:

- Service Common Data Types are as specified in Section 3.1.5
- Service Common Operation Structure represents the high-level decomposition into HEADER / DATA areas
- Operation Common Headers contain the definition of the HEADER area for individual operations

NOTE: We have specified this at the level of both Association and Data Exchange Operations. One of the reasons for doing this is that ALL Association Operation Structures just consist of a HEADER area (and no DATA). We can therefore consider Service Specific information to be only relevant for the Data Exchange Operations.

- Service Specific Data Parameter is a definition of the DATA area required for each Data Exchange Operation
- Service Specific Data Types are those required in the DATA area

The following sections describes these concepts can be used to express the existing services and operations. The operation tables from chapter 2 have been repeated to simplify cross-reference.

4.1.1 Association Operations

This consists of the BIND, UNBIND and PEER-ABORT operations.

4.1.1.1 BIND

The BIND operation parameter table is repeated below:

BIND	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
initiator-identifier	M		M		M		M		M	
responder-identifier		M		M		M		M		M
responder-port-identifier	M		M		M		M		M	
service-type	M		M		M		M		M	
version-number	M	C	M	C	M	C	M	C	M	C
service-instance-identifier	M		M		M		M		M	
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

This ‘fits’ into the proposed Service Common Operation Structure as:

BIND		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	initiator-identifier	M	
	responder-identifier		M
	responder-port-identifier	M	
	service-type	M	
	version-number	M	C
	service-instance-identifier	M	

BIND		NEW service	
	result		M
	diagnostic		C

Therefore the BIND Operation consists entirely of an Operation Common Header (ie no DATA area).

4.1.1.2 UNBIND

The UNBIND operation parameter table is repeated below:

UNBIND	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
Unbind-reason	M		M		M		M		M	
Result		M		M		M		M		M

This 'fits' into the proposed Service Common Operation Structure as:

UNBIND		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	unbind-reason	M	
	result		M

Therefore the UNBIND Operation consists entirely of an Operation Common Header (ie no DATA area).

4.1.1.3 PEER-ABORT

The PEER-ABORT operation parameter table is repeated below:

PEER-ABORT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Diagnostic	M		M		M		M		M	

This 'fits' into the proposed Service Common Operation Structure as:

PEER-ABORT		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	diagnostic	M	

Therefore the PEER-ABORT Operation consists entirely of an Operation Common Header (ie no DATA area).

4.1.2 Data Exchange Operations

This consists of the START, STOP, TRANSFER-DATA, SCHEDULE-STATUS-REPORT, STATUS-REPORT, SYNC-NOTIFY, ASYNC-NOTIFY, THROW-EVENT and GET-PARAMETER operations.

4.1.2.1 START

The START operation parameter table is repeated below:

START	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Invoker-credentials	M		M		M		M		M	
Performer-credentials		M		M		M		M		M
Invoke-ID	M	M	M	M	M	M	M	M	M	M
start-time	M		M		M					
stop-time	M		M		M					
requested-frame-quality	M									
Global-VCID			M		M					
control-word-type					M					
tc-vcid					C					
update-mode					M					
first-cltu-identification							M			
first-packet-identification									M	
start-production-time								C	M	
stop-production-time								C	M	
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

This ‘fits’ into the proposed Service Common Operation Structure as:

START		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	Invoke-ID	M	M
	Result		M
DATA	service-type	M	

START		NEW service	
	Service-dependent-data-parameter[9]	M	C

4.1.2.2 STOP

The STOP operation parameter table is repeated below:

STOP	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
invoke-ID	M	M	M	M	M	M	M	M	M	M
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

This 'fits' into the proposed Service Common Operation Structure as:

STOP		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	Invoke-ID	M	M
	Result		M
DATA	service-type	M	
	Service-dependent-data-parameter[10]	M	C

4.1.2.3 TRANSFER-DATA

The TRANSFER-DATA operation parameter table is repeated below:

TRANSFER-DATA	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
Earth-receive-time	M		M		M					
antenna-ID	M		M		M					
data-link-continuity	M		M		M					

[9] As the Diagnostic parameter is a Service Specific Data Type it is NOT in the HEADER area but becomes part of the Service-dependent-data-parameter

[10] Here the Service-dependent-data-parameter consists of ONLY the Diagnostic parameter that is of a Service Specific Data Type.

TRANSFER-DATA	RAF		RCF		ROCF		CLTU		FSP	
delivered-frame-quality	M									
private-annotation	M		M		M					
performer-credentials								M		M
invoke-ID							M	M	M	M
cltu-identification							M			
earliest-radiation-time							M			
latest-radiation-time							M			
packet-identification									M	M
earliest-production-time									M	
latest-production-time									M	
delay-time							M		M	
Report							M			
Transmission-mode									M	
MAP-identifier									M	
Blocking									M	
processing-started-notification									M	
radited-notification									M	
acknowledged-notification									M	
Data	M		M		M		M		M	
cltu-buffer-available								M		
packet-buffer-available										M
Result								M		M
Diagnostic								C		C

As a consequence of the inherent differences discussed in Section 3.1.4.3 (related to confirmed/unconfirmed operations for Forward/Return Services respectively), there are certain issues that prevent a logical straightforward use of the Service Common Operation Structure:

- The HEADER for a Return Service would ‘only’ require the invoker-credentials parameter whereas for a Forward Service, the HEADER would requires the performer-credentials/Invoke-ID parameters in addition to the invoker-credentials.
- There is (by definition of an unconfirmed operation) no RESULT parameters for Return Services TRANSFER-DATA operation.

It is certainly possible that we could specify the HEADER to include all the parameters required by a Forward Service and then stipulate that a Return Service only makes use of the invoker-credentials parameter (ignoring the others or setting them to a default ‘null’ value).

However, this is not the preferred option as it is not very flexible or extendable.

It is therefore proposed that the TRANSFER-DATA operation is split into a RET-TRANSFER-DATA operation (for the Return services - which is an unconfirmed operation) and FWD-TRANSFER-DATA operation (for the Forward Services - which is a confirmed operation).

This would give rise to:

RET-TRANSFER-DATA		NEW RETURN service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
DATA	service-type[11]	M	
	Service-dependent-data-parameter	M	

FWD-TRANSFER-DATA		NEW FORWARD service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	Invoke-ID	M	M
	Return		M
DATA	service-type[12]	M	M
	Service-dependent-data-parameter	M	M

It can be argued that this separation is also driven by the protocol/delivery differences between Forward/Return Services for the TRANSFER-DATA operation. In simple terms, the major difference is that Return Service providers use a 'transfer-buffer' that allows multiple TRANSFER-DATA and SYNC-NOTIFY operations to be mapped to a single PDU. The rational for this approach is described in more detail in the Return Service CCSDS Recommendation documents.

Obviously this is an issue that would need to be discussed at a CCSDS Meeting for agreement/approval, as it is a change to the CCSDS Recommendations.

4.1.2.4 SCHEDULE-STATUS-REPORT

The SCHEDULE-STATUS-REPORT operation parameter table is repeated below:

SCHEDULE-STATUS-REPORT	RAF	RCF	ROCF	CLTU	FSP

[11] Limited to Return Services – RAF, RCF and ROCF

[12] Limited to Forward Service – CLTU and FSP

SCHEDULE-STATUS-REPORT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
Invoke-ID	M	M	M	M	M	M	M	M	M	M
report-request-type	M		M		M		M		M	
reporting-cycle	C		C		C		C		C	
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

This 'fits' into the proposed Service Common Operation Structure as:

SCHEDULE-STATUS-REPORT		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	Invoke-ID	M	M
	report-request-type	M	
	reporting-cycle	C	
	Result		M
	Diagnostic		C

Therefore the SCHEDULE-STATUS-REPORT Operation consists entirely of an Operation Common Header (ie no DATA area). This is the only Data Exchange Operation for which this is the case.

4.1.2.5 STATUS-REPORT

The SCHEDULE-STATUS-REPORT operation parameter table is repeated below:

STATUS-REPORT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
Invoker-credentials	M		M		M		M		M	
number-of-error-free-frames-delivered	M									
number-of-frames-delivered	M		M							
number-of-frames-processed					M					
number-of-ocfs-delivered					M					
frame-sync-lock-status	M		M		M					
symbol-sync-lock-status	M		M		M					
Subcarrier-lock-status	M		M		M					

STATUS-REPORT	RAF		RCF		ROCF		CLTU		FSP	
carrier-lock-status	M		M		M					
cltu-last-processed							M			
cltu-last-OK							M			
cltu-status							C			
radiation-start-time							C			
radiation-stop-time							C			
packet-identification-last-processed									M	
production-start-time									C	
packet-status									C	
packet-identification-last-OK									M	
production-stop-time									C	
production-status	M		M		M		M		M	
uplink-status							M			
number-of-cltus-received							M			
number-of-cltus-processed							M			
number-of-cltus-radiated							M			
cltu-buffer-available							M			
number-of-packets-received							M			
number-of-packets-processed							M			
number-of-packets-radiated							M			
number-of-packets-acknowledged							M			
packet-buffer-available							M			

This 'fits' into the proposed Service Common Operation Structure as:

STATUS-REPORT		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
DATA	service-type	M	
	Service-dependent-data-parameter	M	
	Service-dependent-Report		

4.1.2.6 NOTIFY OPERATIONS

There are TWO notification events described in the CCSDS Recommendation documents, SYNC-NOTIFY used by RETURN Services and ASYNC-NOTIFY used by FORWARD Services. They are both invoked by the SLE Service Provider to notify the User of an event affecting production or provision of the service.

4.1.2.6.1 SYNC-NOTIFY

The SYNC-NOTIFY operation parameter table is repeated below:

SYNC-NOTIFY	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**		**I**		**I**		**I**		**I**	
invoker-credentials	M		M		M					
notification-type	M		M		M					
notification-value	C		C		C					

This 'fits' into the proposed Service Common Operation Structure as:

SYNC-NOTIFY		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
DATA	service-type	M	
	Service-dependent-data-parameter	M	

4.1.2.6.2 ASYNC-NOTIFY

The ASYNC-NOTIFY operation parameter table is repeated below:

ASYNC-NOTIFY	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials							M		M	
notification-type							M		M	
directive-executed-identification									C	
event-thrown-identification							C		C	
cltu-last-processed							M			
cltu-last-OK							M			
cltu-status							M			
radiation-start-time							C			
radiation-stop-time							C			
packet-identification-list									C	
fop-alert									C	
packet-identification-last-processed									M	
production-start-time									C	
packet-status									C	
packet-identification-last-ok									M	
production-stop-time									C	
production-status							M		M	

ASYNC-NOTIFY	RAF		RCF		ROCF		CLTU		FSP	
uplink-status							M			

This 'fits' into the proposed Service Common Operation Structure as:

ASYNC-NOTIFY		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
DATA	service-type	M	
	Service-dependent-data-parameter	M	

As can be seen from above, the SYNC-NOTIFY and ASYNC-NOTIFY operations are of a similar structure when considering the proposed Service Common Operation Structure. However, the two operations remain separate because of the differences in protocol/delivery (see CCSDS Recommendation documents for details).

4.1.2.7 THROW-EVENT

The THROW-EVENT operation parameter table is repeated below:

THROW-EVENT	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials							M		M	
performer-credentials								M		M
invoke-ID							M	M	M	M
event-invocation-identification							M	M	M	M
event-identifier							M		M	
event-qualifier							M		M	
result								M		M
diagnostic								C		C

This 'fits' into the proposed Service Common Operation Structure as:

THROW-EVENT		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	invoke-ID	M	M
	Result		M
	Diagnostic		C
DATA	service-type	M	M

THROW-EVENT			NEW service	
	Service-dependent-data-parameter		M	M
		event-identifier		
		event-qualifier		
		event-invocation-identification		

4.1.2.8 GET-PARAMETER

The GET-PARAMETER operation parameter table is repeated below:

GET-PARAMETER	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials	M		M		M		M		M	
performer-credentials		M		M		M		M		M
invoke-ID	M	M	M	M	M	M	M	M	M	M
raf-parameter	M	C								
rcf-parameter			M	C						
rocf-parameters					M	C				
cltu-parameter							M	C		
fsp-parameter									M	C
parameter-value		C		C		C		C		C
Result		M		M		M		M		M
Diagnostic		C		C		C		C		C

This ‘fits’ into the proposed Service Common Operation Structure as:

GET-PARAMETER		NEW service	
Structure	**Parameters**	**I**	**R**
HEADER	invoker-credentials	M	
	performer-credentials		M
	invoke-ID	M	M
	Result		M
	Diagnostic		C
DATA	service-type	M	
	Service-dependent-data-parameter	M	C
	parameter-identifier		
	parameter-value		

4.1.2.9 FSP-INVOKE-DIRECTIVE

The INVOKE-DIRECTIVE operation parameter table is repeated below:

INVOKE-DIRECTIVE	RAF		RCF		ROCF		CLTU		FSP	
Parameters	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**	**I**	**R**
invoker-credentials									M	
performer-credentials										M
invoke-ID									M	M
directive-identification									M	M
directive									M	
Result										M
Diagnostic										C

This 'fits' into the proposed Service Common Operation Structure as:

INVOKE-DIRECTIVE			NEW service	
Structure	**Parameters**		**I**	**R**
HEADER	invoker-credentials		M	
	performer-credentials			M
	invoke-ID		M	M
	Result			M
DATA	service-type		M	
	Service-dependent-data-parameter		M	M
		directive-identification		
		directive		
		Dianostic		

4.1.3 Summary

From the tables above in Sections 4.1.1 and 4.1.2, we can see that all current Service operations can be represented using the proposed Generic Service Common Operation Structure (see Table 4-1).

There is one issue that would require clarification/agreement/approval:

For the TRANSFER-DATA operation, whether to keep the existing operation or split it into a RET-TRANSFER-DATA and FWD-TRANSFER-DATA.

One additional possibility is whether a set of 'standard' HEADER definitions can be identified to further increase the commonality. It would seem logical that 'confirmed operations' would have a different signature/structure from 'unconfirmed operations' (because of the performer-credentials, Result and Diagnostic parameters). The idea would be to have for example a 'Primary Header':

Confirmed Operation	
Structure	**Parameters**
HEADER	invoker-credentials
	performer-credentials
	invoke-ID
	Result
	Diagnostic

Unconfirmed Operation	
Structure	**Parameters**
HEADER	invoker-credentials

If we review the proposed Header definitions for the individual operations against this, we notice that there are a number of 'special cases' that can be identified:

- Operation specific parameters in the Header (extend with concept of 'Secondary Header')
- The BIND and UNBIND operations (confirmed) do not require the invoke-ID
- The UNBIND operation does not include the Diagnostic parameter
- The PEER-ABORT operation is significantly different from the other operations (no credential parameters and consists of just a Diagnostic parameter)
- For some confirmed operations the Diagnostic parameter contains Service Specific information and is therefore part of the service-dependent-data-parameter and not included in the Header area

It would appear that the Data Exchange Operations fit this 'standard Header' concept with less 'special cases' than the Association Operations (which have very distinct Header parameters). It is possible that we could 'force' this commonality on operations and just ignore the parameters that are not required by that operation. However, as the proposed Generic Service Common Operation Structure is

a very flexible solution (particularly important for the definition of NEW Services, including the definition of NEW operations), it is not entirely clear that this would be necessary. It is felt that this issue be reviewed later during the design/implementation/prototyping phases.

4.2 Service Provider State Transitions

Comparing the tables in Section 3.2 it is clear that the state transitions for a Forward Service Provider are considerably different from those of a Return Service Provider. Some of the reasons for this are:

1. The fact that for a Forward Service the CCSDS Recommendation documents specify that the Association operations BIND and UNBIND are only invoked by the User. Therefore the Return events for these operations are not applicable for a Forward Service Provider. This is not the case for the Return Service Provider as the provider can initiate a BIND.

   The issue of whether to restrict the BIND/UNBIND operations to Service Users only, has been mentioned previously. Here it would obviously result in a more generic (and simpler) State Transition machine.

2. The inherent differences in the DataTransfer operation lead to Service Specific behaviour, especially in the Active State. This covers issues like

   - The delivery mode for Return Services (online/offline) compared to the AD/BD mode for Forward FSP
   - The use of the Data Transfer buffer - for Return Services this is used to transmit data back to the User whereas for Forward Services it is used to store data for the Service Provider.
   - The detailed functional behaviour of the provider during service production. In particular, the Forward Services have Service Specific behaviour related to the Production Status.

3. The Return Services do not require the THROW-EVENT and INVOKE-DIRECTIVE operations.

   Whilst the current Return Services do not utilise/require the above operations (and only FSP uses INVOKE-DIRECTIVE), it is not known whether Future Return Services would. It is therefore worth considering the option to include these operations if it would lead to a more generic State Transition machine. The option to allow a Future Service to ignore these operations if they are not required should be included.

   There is one issue of caution with the INVOKE-DIRECTIVE operation, which is currently used by the FSP Service (only) for TC directives to cause a (re-)configuration of the TC protocol. It would be possible that the INVOKE-DIRECTIVE operation could be (mis-)used to reconfigure equipment at a Ground Station. It should therefore be explicitly described in the CCSDS Recommendation document the extent to which this is allowed.

There are however a few commonalities as can be seen from the ScheduleStatusReport (FGEN-7 / RGEN-22), ‘Reporting-Cycle Timer’ (FGEN-8 / RGEN-23), GetParameter (FGEN-10 / RGEN-24), PeerAbort(FGEN-21 / RGEN-25), ‘invalid PDU’ (FGEN-24 / RGEN-26) and ‘not authenticated PDU’ (FGEN-26 / RGEN-29) events.

We therefore restrict the commonality and consider the Return Services separately from the Forward Services.

4.2.1 Return Services

In Section 3.2.2 we have seen that for the Return Services the behaviour is completely common. It is therefore conceivable that a single State Machine could be defined to represent all Return Services.

4.2.2 Forward Services

The deltas between the association handling in FSP and CLTU are in essence a result of the fact that in the case of FSP, several users may share the same service production, while in CLTU there is a one-to-one relationship. As a consequence, for FSP, the requirement was raised that a transition of the production status to operational shall only be reported to those users that have been made aware of a different productions status value before. For that reason, the FSP Service needs to do some additional book keeping which in the state machine appears as additional conditional tests e.g. for the BIND operation the predicate "production configured" is tested and "notify production operational" may be updated in the course of processing the BIND invocation. The more general case is FSP (several users sharing the same service production) and for the generic framework we should not constrain ourselves to the simplified CLTU case. Although not strictly required in the CLTU case, the additional predicates processing could be done in CLTU as well and this way the processing of the operation can be made common. In fact, the same consideration can be applied to the Return Services. For those services, the production status never is 'configured' and the associated branch in the state table will never be executed. But its presence does not harm either and therefore the operation can be made truly identical for all services.

The difference in processing of the UNBIND is in part attributable to the additional predicates as discussed above. The stopping of the return-timeout-period timers should be added for CLTU in order to leave the state machine in a well-defined state.

The difference in the START operation is again only necessary in order not to duplicate any production status notifications to the user. As for the BIND, it should be possible to modify CLTU accordingly without actually affecting the externally observable behaviour.

The differences in the TRANSFER-DATA operation are attributable to the different way of handling the AD and BD frames that FSP must support and this difference cannot be hidden.

The 'production interrupted' event has to be handled differently in CLTU, as the service may be run in the mode where the CLTU buffer is not cleared after STOP in order to achieve a kind of offline commanding capability. However, the proper way to achieve this is to truly specify an offline forward service. If that were done, the FSP way of processing the event would also apply to the CLTU service. This same consideration applies to the different handling for the 'production halted' and 'protocol abort' events.

Appendix A - State transition for RAF SERVICE PROVIDER

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1
3	(-rafBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
4	(+rafBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE [ignore] ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
5	(rafBindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+rafBindReturn) → 2 ELSE (-rafBindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period'} → 1
7	(rafUnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
8	(rafUnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1
9	(rafStartInvocation)	[ignore]	IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+rafStartReturn) → 3 initialize transfer buffer ELSE (-rafStartReturn) → 2	{peer abort 'protocol error'} → 1
10	(rafStopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {transmit buffer} (+rafStopReturn) ELSE (+rafStopReturn) ELSE (-rafStopReturn) → 3
11	(rafStopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {pass buffer contents} (+rafStopReturn) ELSE (+rafStopReturn) ELSE (-rafStopReturn) → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
12	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} ELSE {insert annotated frame} → 3
13	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame}→ 3 {start release timer} ELSE {insert annotated frame} → 3
14	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {insert annotated frame} {start release timer} ELSE {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame} → 3 {start release timer} ELSE {insert annotated frame} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
15	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {start release timer} ELSE [ignore]
16	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
17	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE → 3 {sync notify 'end of data'} {transmit buffer}
18	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
19	'loss of frame synchronization', "timely online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {start release timer} {sync notify 'loss of frame sync'} ELSE {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
20	'loss of frame synchronization', "complete online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
21	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
22	(rafScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+rafScheduleStatusReport) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 2 (-rafScheduleStatusReportReturn)	IF "positive result" THEN (+rafScheduleStatusReport) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 3 (-rafScheduleStatusReportReturn)
23	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
24	(rafGetParameterInvocation)	[ignore]	IF "positive result" THEN (+rafGetParameterReturn)→ 2 ELSE (-rafGetParameterReturn) → 2	IF "positive result" THEN (+rafGetParameterReturn)→ 3 ELSE (-rafGetParameterReturn) → 3
25	(rafPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
26	'invalid protocol data unit'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
27	'return SLE-PDU with unsolicited Invoke-ID'	[ignore]	{peer abort ('unsolicited Invoke-ID')} → 1	{peer abort ('unsolicited Invoke-ID')} → 1
28	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
29	'not authenticated SLE-PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3

Appendix B - State transition for RCF SERVICE PROVIDER

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1
3	(-rcfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
4	(+rcfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE [ignore] ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
5	(rcfBindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+rcfBindReturn) → 2 ELSE (-rcfBindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period'} → 1
7	(rcfUnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
8	(rcfUnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1
9	(rcfStartInvocation)	[ignore]	IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+rcfStartReturn) → 3 initialize transfer buffer ELSE (-rcfStartReturn) → 2	{peer abort 'protocol error'} → 1
10	(rcfStopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {transmit buffer} (+rcfStopReturn) ELSE (+rcfStopReturn) ELSE (-rcfStopReturn) → 3
11	(rcfStopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {pass buffer contents} (+rcfStopReturn) ELSE (+rcfStopReturn) ELSE (-rcfStopReturn) → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
12	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} ELSE {insert annotated frame} → 3
13	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame} → 3 {start release timer} ELSE {insert annotated frame} → 3
14	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {increment buffer size by one} {sync notify 'data discarded'} {insert annotated frame} {start release timer} ELSE {insert annotated frame} {start release timer} ELSE IF "buffer empty" THEN {insert annotated frame} → 3 {start release timer} ELSE {insert annotated frame} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
15	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN {increment buffer size by one} {sync notify 'data discarded'} {start release timer} ELSE [ignore]
16	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
17	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE → 3 {sync notify 'end of data'} {transmit buffer}
18	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
19	'loss of frame synchronization', "timely online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {start release timer} {sync notify 'loss of frame sync'} ELSE {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
20	'loss of frame synchronization', "complete online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
21	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
22	(rcfScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+rcfScheduleStatusReport) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 2 (-rcfScheduleStatusReportReturn)	IF "positive result" THEN (+rcfScheduleStatusReport) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE → 3 (-rcfScheduleStatusReportReturn)
23	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
24	(rcfGetParameterInvocation)	[ignore]	IF "positive result" THEN (+rcfGetParameterReturn)→ 2 ELSE (-rcfGetParameterReturn) → 2	IF "positive result" THEN (+rcfGetParameterReturn)→ 3 ELSE (-rcfGetParameterReturn) → 3
25	(rcfPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
26	'invalid SLE-PDU'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
27	'return SLE-PDU with unsolicited Invoke-ID'	[ignore]	{peer abort ('unsolicited Invoke-ID')} → 1	{peer abort ('unsolicited Invoke-ID')} → 1
28	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
29	'not authenticated SLE-PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3

Appendix C - State transition for ROCF SERVICE PROVIDER

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1
3	(-rocfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
4	(+rocfBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE [ignore] ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
5	(rocfBindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+rocfBindReturn) → 2 ELSE (-rocfBindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period'} → 1
7	(rocfUnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
8	(rocfUnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1
9	(rocfStartInvocation)	[ignore]	IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+rocfStartReturn) → 3 initialize transfer buffer ELSE (-rocfStartReturn) → 2	{peer abort 'protocol error'} → 1
10	(rocfStopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF NOT "buffer empty" THEN {transmit buffer} (+rocfStopReturn) ELSE (+rocfStopReturn) ELSE (-rocfStopReturn) → 3
11	(rocfStopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "positive result" THEN → 2 IF NOT "buffer empty" THEN {pass buffer contents} (+rocfStopReturn) ELSE (+rocfStopReturn) ELSE (-rocfStopReturn) → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
12	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated OCF} ELSE {insert annotated OCF} → 3
13	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated OCF} {start release timer} ELSE IF "buffer empty" THEN {insert annotated OCF} → 3 {start release timer} ELSE {insert annotated OCF} → 3
14	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {insert annotated OCF} {start release timer} ELSE {insert annotated OCF} {start release timer} ELSE IF "buffer empty" THEN {insert annotated OCF} → 3 {start release timer} ELSE {insert annotated OCF} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
15	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {start release timer} ELSE [ignore]
16	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
17	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}
18	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
19	'loss of frame synchronization', "timely online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {start release timer} {sync notify 'loss of frame sync'} ELSE {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE {sync notify 'loss of frame sync'} → 3
20	'loss of frame synchronization', "complete online" delivery mode	Not applicable	[ignore] → 2	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'loss of frame sync'} {start release timer} ELSE IF "buffer empty" THEN {sync notify 'loss of frame sync'} → 3 {start release timer} ELSE (sync notify 'loss of frame sync'} → 3
21	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
22	(rocfScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+rocfScheduleStatusReport) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-rocfScheduleStatusReport Return) → 2	IF "positive result" THEN (+rocfScheduleStatusReport) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-rocfScheduleStatusReport Return) → 3
23	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
24	(rocfGetParameterInvocation)	[ignore]	IF "positive result" THEN (+rocfGetParameterReturn) → 2 ELSE (-rocfGetParameterReturn) → 2	IF "positive result" THEN (+rocfGetParameterReturn) → 3 ELSE (-rocfGetParameterReturn) → 3
25	(rocfPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
26	'invalid protocol data unit'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
27	'return SLE-PDU with unsolicited Invoke-ID'	[ignore]	{peer abort ('unsolicited Invoke-ID')} → 1	{peer abort ('unsolicited Invoke-ID')} → 1
28	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
29	'not authenticated SLE-PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3

Appendix D - State transition for GENERIC DATA TRANSFER SERVICE PROVIDER

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
1	'start of service instance provision period'	IF "provider initiated" THEN {invoke bind} → 1 ELSE [ignore] → 1	Not applicable	Not applicable
2	'return <n> timer expired'	IF "bind pending" THEN {return timeout} → 1 IF "provision period" THEN {invoke bind} ELSE [ignore] ELSE Not applicable → 1	{peer abort 'return timeout'} → 1	{peer abort 'return timeout'} → 1
3	(-genericBindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 1 stop return <n> timer IF "retry permitted" THEN {invoke bind} ELSE release resources ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
4	(+**generic**BindReturn)	IF "bind pending" THEN set "bind pending" FALSE → 2 stop return <n> timer IF NOT "compatible" THEN {invoke unbind} ELSE {invoke start} ELSE [ignore] → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
5	(**generic**BindInvocation)	IF "provider initiated" THEN [ignore] → 1 ELSE IF "positive result" THEN (+**generic**BindReturn) → 2 ELSE (-**generic**BindReturn) → 1	{peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
6	'end of service instance provision period'	[ignore]	IF "provider initiated" THEN {invoke unbind} → 2 ELSE {peer abort 'end of service instance provision period'} → 1	{peer abort 'end of service instance provision period' } → 1
7	(**generic**UnbindReturn)	[ignore]	IF "unbind pending" THEN {provider unbind} → 1 IF "done" THEN release resources ELSE [ignore] ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
8	(**generic**UnbindInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE {user unbind} → 1 IF "end" THEN release resources ELSE [ignore]	{peer abort 'protocol error'} → 1
9	(-**generic**StartReturn)	[ignore]	IF "start pending" THEN set "start pending" FALSE → 2 stop return <n> timer IF "retry permitted" THEN {invoke start} ELSE [ignore] → 2 ELSE {peer abort 'protocol error'} → 1	{peer abort 'protocol error'} → 1
10	(+**generic**StartReturn)	[ignore]	initialize transfer buffer → 3	{peer abort 'protocol error'} → 1
11	(**generic**StartInvocation)	[ignore]	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE IF "unbind pending" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN (+**generic**StartReturn) → 3 initialize transfer buffer ELSE (-**generic**StartReturn) → 2	{peer abort 'protocol error'} → 1
12	(-**generic**StopReturn)	[ignore]	{peer abort 'protocol error'} → 1	IF "stop pending" THEN set "stop pending" FALSE → 3 stop return <n> timer ELSE {peer abort 'protocol error'} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
13	(+**generic**StopReturn)	[ignore]	{peer abort 'protocol error'} → 1	IF "stop pending" THEN {provider stop} → 2 ELSE {peer abort 'protocol error'} → 1
14	(**generic**StopInvocation) "complete online" or "offline" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {transmit buffer} (+**generic**StopReturn) ELSE (+**generic**StopReturn) ELSE (-**generic**StopReturn) → 3
15	(**generic**StopInvocation) "timely online" delivery mode	[ignore]	{peer abort 'protocol error'} → 1	IF "provider initiated" THEN {peer abort 'protocol error'} → 1 ELSE IF "positive result" THEN → 2 IF (NOT "buffer empty") THEN {pass buffer contents} (+**generic**StopReturn) ELSE (+**generic**StopReturn) ELSE (-**generic**StopReturn) → 3
16	'data available', "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated data} ELSE {insert annotated data} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
17	'data available', "complete online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {insert annotated data} {start release timer} ELSE IF "buffer empty" THEN {insert annotated data} → 3 {start release timer} ELSE {insert annotated data} → 3
18	'data available', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {insert annotated data} {start release timer} ELSE {insert annotated data} {start release timer} ELSE IF "buffer empty" THEN {insert annotated data} → 3 {start release timer} ELSE {insert annotated data} → 3

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
19	'release timer expired', "timely online" delivery mode	Not applicable	Not applicable	{pass buffer contents}→ 3 IF "congested" THEN increment buffer size by one {sync notify 'data discarded'} {start release timer} ELSE [ignore]
20	'release timer expired', "complete online" delivery mode	Not applicable	Not applicable	{transmit buffer} → 3
21	'end of data', "timely online" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {pass buffer contents} → 3 IF "congested" THEN {sync notify 'data discarded'} {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} {transmit buffer} ELSE → 3 {sync notify 'end of data'} {transmit buffer}
22	'end of data', "complete online" delivery mode or "offline" delivery mode	Not applicable	Not applicable	IF "buffer full" THEN {transmit buffer} → 3 {sync notify 'end of data'} {transmit buffer} ELSE {sync notify 'end of data'} → 3 {transmit buffer}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
23	'production status change', "timely online" delivery mode or "complete online" delivery mode	Not applicable	IF NOT "unbind pending" THEN {sync notify 'production status change'} → 2 ELSE [ignore] → 2	{sync notify 'production status change'} → 3
24	(genericSchedStatusRepInvocation)	[ignore]	IF "positive result" THEN (+genericSchedStatusRep) → 2 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE {stop reporting-cycle timer} ELSE → 2 (-genericSchedStatusRepReturn)	IF "positive result" THEN (+genericSchedStatusRep) → 3 IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE {stop reporting-cycle timer} ELSE → 3 (-genericSchedStatusRepReturn)
25	'reporting-cycle timer expired'	Not applicable	{periodic report} → 2	{periodic report} → 3
26	(genericGetParameterInvocation)	[ignore]	IF "positive result" THEN (+genericGetParameterReturn) → 2 ELSE (-genericGetParameterReturn) → 2	IF "positive result" THEN (+genericGetParameterReturn) → 3 ELSE (-genericGetParameterReturn) → 3
27	(genericThrowEventInvocation)	[ignore]	IF "positive result" THEN (+genericThrowEventReturn) forward event to Complex Management ELSE (-genericThrowEventReturn)	IF "positive result" THEN (+genericThrowEventReturn) forward event to Complex Management ELSE (-genericThrowEventReturn)
28	'action list completed'	Not applicable	{sync notify 'action list completed'}	{sync notify 'action list completed'}
29	'action list not completed'	Not applicable	{sync notify 'action list not completed'}	{sync notify 'action list not completed'}
30	'event condition evaluated to false'	Not applicable	{sync notify 'event condition evaluated to false'}	{sync notify 'event condition evaluated to false'}
31	(genericPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
32	'invalid PDU'	[ignore]	{peer abort 'encoding error'} → 1	{peer abort 'encoding error'} → 1
33	'return PDU with unsolicited Invoke-ID"	[ignore]	{peer abort 'unsolicited Invoke-ID' } → 1	{peer abort 'unsolicited Invoke-ID' } → 1
34	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
35	'not authenticated PDU'	[ignore] → 1	[ignore] → 2	[ignore] → 3

Appendix E - State transition for CLTU SERVICE PROVIDER

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
1.	(cltuBindInvocation)	IF "positive result" THEN (+cltuBindReturn) → 2 ELSE (-cltuBindReturn)	IF "same service instance" THEN (-cltuBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1	IF "same service instance" THEN (-cltuBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1
2.	'end of service instance provision period'	{clean up}	{peer abort ('end of service instance provision period')} → 1	{peer abort ('end of service instance provision period')} → 1
3.	(cltuUnbindInvocation)	[ignore]	(cltuUnbindReturn) → 1 stop reporting-cycle timer IF "end" THEN release resources ELSE [ignore]	{peer abort ('protocol error')} → 1
4.	(cltuStartInvocation)	[ignore]	IF "positive result" THEN (+cltuStartReturn) → 3 ELSE (-cltuStartReturn)	{peer abort ('protocol error')} → 1
5.	(cltuStopInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" THEN {initiate stop} → 2 ELSE (-cltuStopReturn)
6.	(cltuTransferDataInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") THEN buffer CLTU (+cltuTransferDataReturn) ELSE discard CLTU (-cltuTransferDataReturn)

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
7.	(cltuScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+cltuScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-cltuScheduleStatusReportReturn)	IF "positive result" THEN (+cltuScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-cltuScheduleStatusReportReturn)
8.	'reporting-cycle timer expired'	Not applicable	{periodic report}	{periodic report}
9.	'return-timeout-period timer <n> expired'	Not applicable	{peer abort ('return timeout')} → 1	{peer abort ('return timeout') → 1
10.	(cltuGetParameterInvocation)	[ignore]	IF "positive result" THEN (+cltuGetParameterReturn) ELSE (-cltuGetParameterReturn)	IF "positive result" THEN (+cltuGetParameterReturn) ELSE (-cltuGetParameterReturn)
11.	(cltuThrowEventInvocation)	[ignore]	IF "positive result" THEN (+cltuThrowEventReturn) forward event to Complex Management ELSE (-cltuThrowEventReturn)	IF "positive result" THEN (+cltuThrowEventReturn) forward event to Complex Management ELSE (-cltuThrowEventReturn)
12.	'cltu radiated'	[ignore]	IF "report" THEN {notify('cltu radiated')} ELSE [ignore]	IF "report" THEN {notify('cltu radiated')} ELSE [ignore]
13.	'sldu expired'	IF "continue" THEN clear CLTU buffer ELSE [ignore]	Not applicable	{notify('sldu expired') and block}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
14.	'production interrupted'	IF "continue" THEN clear CLTU buffer ELSE [ignore]	{notify('production interrupted') and clear}	{notify('production interrupted') and block} set "notify production operational" to TRUE
15.	'production halted'	IF "continue" THEN clear CLTU buffer ELSE [ignore]	{notify('production halted') and clear} set "notify production operational" to TRUE	{notify('production halted') and block} set "notify production operational" to TRUE
16.	'production operational'	[ignore]	IF "notify production operational" THEN {notify('production operational')} set "notify production operational" to FALSE	IF "notify production operational" THEN {notify('production operational')} set "notify production operational" to FALSE
17.	'buffer empty'	[ignore]	Not applicable	{notify('buffer empty')}
18.	'action list completed'	Not applicable	{notify('action list completed')}	{notify('action list completed')}
19.	'action list not completed'	Not applicable	{notify('action list not completed')}	{notify('action list not completed')}
20.	'event condition evaluated to false'	Not applicable	{notify('event condition evaluated to false')}	{notify('event condition evaluated to false')}
21.	(cltuPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
22.	'protocol abort'	[ignore]	{clean up} → 1	IF "continue" THEN stop reporting-cycle timer → 1 ELSE {clean up} → 1
23.	'unsolicited invoke-ID'	[ignore]	{peer abort ('unsolicited invoke-ID')} → 1	{peer abort ('unsolicited invoke-ID')} → 1
24.	'invalid SLE-PDU'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
25.	'unexpected SLE-PDU'	[ignore]	{peer abort ('protocol error')} → 1	{peer abort ('protocol error')} → 1
26.	'not authenticated SLE-PDU'	[ignore]	[ignore]	[ignore]

Appendix F – State transition for FSP SERVICE PROVIDER

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
1.	(fspBindInvocation)	IF "positive result" .AND. "production configured" THEN {accept bind} → 2 ELSE IF "positive result" THEN (+fspBindReturn) → 2 ELSE (-fspBindReturn)	IF "same service instance" THEN (-fspBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1	IF "same service instance" THEN (-fspBindReturn ('already bound')) ELSE {peer abort ('protocol error')} → 1
2.	'end of service instance provision period'	[ignore]	{peer abort ('end of service instance provision period')} → 1	{peer abort ('end of service instance provision period')} → 1
3.	(fspUnbindInvocation)	[ignore]	(fspUnbindReturn) → 1 stop reporting-cycle timer stop all return-timeout-period timers set "notify production operational" to FALSE IF "end" THEN release resources ELSE [ignore]	{peer abort ('protocol error')} → 1
4.	(fspStartInvocation)	[ignore]	IF "positive result" THEN (+fspStartReturn) → 3 ELSE IF "production off" THEN {reject start} ELSE (-fspStartReturn)	{peer abort ('protocol error')} → 1

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
5.	(fspStopInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" THEN {initiate stop} → 2 ELSE (-fspStopReturn)
6.	(fspTransferDataInvocation) with BD Packet	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") THEN queue packet (+fspTransferDataReturn) ELSE discard packet (-fspTransferDataReturn)
7.	(fspTransferDataInvocation) with AD Packet and unblock AD	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") .AND. ("AD blocked") THEN queue packet Set "AD blocked" to FALSE (+fspTransferDataReturn) ELSE discard packet (-fspTransferDataReturn)
8.	(fspTransferDataInvocation) with AD Packet and .NOT. unblock AD	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" .AND. (.NOT. "service instance blocked") .AND. (.NOT. "AD blocked") THEN queue packet (+fspTransferDataReturn) ELSE discard packet (-fspTransferDataReturn)

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
9.	(fspScheduleStatusReportInvocation)	[ignore]	IF "positive result" THEN (+fspScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-fspScheduleStatusReportReturn)	IF "positive result" THEN (+fspScheduleStatusReportReturn) IF "immediately" THEN {immediate report} ELSE IF "periodically" THEN {periodic report} ELSE stop reporting-cycle timer ELSE (-fspScheduleStatusReportReturn)
10.	'reporting-cycle timer expired'	Not applicable	{periodic report}	{periodic report}
11.	'return-timeout-period timer <n> expired'	Not applicable	{peer abort ('return timeout')} → 1	{peer abort ('return timeout')} → 1
12.	(fspGetParameterInvocation)	[ignore]	IF "positive result" THEN (+fspGetParameterReturn) ELSE (-fspGetParameterReturn)	IF "positive result" THEN (+fspGetParameterReturn) ELSE (-fspGetParameterReturn)
13.	(fspThrowEventInvocation)	[ignore]	IF "positive result" THEN (+fspThrowEventReturn) forward event to Complex Management ELSE (-fspThrowEventReturn)	IF "positive result" THEN (+fspThrowEventReturn) forward event to Complex Management ELSE (-fspThrowEventReturn)
14.	(fspInvokeDirectiveInvocation)	[ignore]	{peer abort ('protocol error')} → 1	IF "positive result" THEN queue directive (+fspInvokeDirectiveReturn) ELSE (-fspInvokeDirectiveReturn)
15.	'packet processing started'	Not applicable	Not applicable	IF "report processing" THEN {notify ('packet processing started')} ELSE [ignore]

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
16.	'packet acknowledged'	Not applicable	IF "report acknowledgement" THEN {notify ('packet acknowledged')} ELSE [ignore]	IF "report acknowledgement" THEN {notify ('packet acknowledged')} ELSE [ignore]
17.	'packet radiated'	Not applicable	IF "report radiation" THEN {notify ('packet radiated')} ELSE [ignore]	IF "report radiation" THEN {notify ('packet radiated')} ELSE [ignore]
18.	'sldu expired'	Not applicable	Not applicable	{notify ('sldu expired') and block}
19.	'packet transmission mode mismatch'	Not applicable	Not applicable	{notify ('packet transmission mode mismatch') and block AD}
20.	'transmission mode capability change'	Not applicable	{notify ('transmission mode capability change')}	{notify ('transmission mode capability change')}
21.	'VC aborted'	Not applicable	{notify ('VC aborted')}	{notify ('VC aborted')}
22.	'production interrupted'	Not applicable	Not applicable	{notify ('production interrupted') and block} set "notify production operational" to TRUE
23.	'production halted'	Not applicable	{notify ('production halted')} set "notify production operational" to TRUE	{notify ('production halted') and block} set "notify production operational" to TRUE
24.	'production operational'	Not applicable	IF "notify production operational" THEN {notify ('production operational')} set "notify production operational" to FALSE	IF "notify production operational" THEN {notify ('production operational')} set "notify production operational" to FALSE
25.	'buffer empty'	Not applicable	{notify ('buffer empty')}	{notify ('buffer empty')}
26.	'no invoke directive capability on this VC'	Not applicable	{notify ('no invoke directive capability on this VC')}	{notify ('no invoke directive capability on this VC')}
27.	'invoke directive capability on this VC established'	Not applicable	{notify ('invoke directive capability on this VC established')}	{notify ('invoke directive capability on this VC established')}
28.	'positive confirm response to directive'	Not applicable	{notify ('positive confirm response to directive')}	{notify ('positive confirm response to directive')}
29.	'negative confirm response to directive'	Not applicable	{notify ('negative confirm response to directive')}	{notify ('negative confirm response to directive')}

No.	Incoming Event	Unbound (State 1)	Ready (State 2)	Active (State 3)
30.	'action list completed'	Not applicable	{notify ('action list completed')}	{notify ('action list completed')}
31.	'action list not completed'	Not applicable	{notify ('action list not completed')}	{notify ('action list not completed')}
32.	'event condition evaluated to false'	Not applicable	{notify ('event condition evaluated to false')}	{notify ('event condition evaluated to false')}
33.	(fspPeerAbortInvocation)	[ignore]	{clean up} → 1	{clean up} → 1
34.	'protocol abort'	[ignore]	{clean up} → 1	{clean up} → 1
35.	'unsolicited invoke-ID'	[ignore]	{peer abort ('unsolicited invoke-ID')} → 1	{peer abort ('unsolicited invoke-ID')} → 1
36.	'invalid SLE-PDU'	[ignore]	{peer abort ('encoding error')} → 1	{peer abort ('encoding error')} → 1
37.	'unexpected SLE-PDU'	[ignore]	{peer abort ('protocol error')} → 1	{peer abort ('protocol error')} → 1
38.	'not authenticated SLE-PDU'	[ignore]	[ignore]	[ignore]