Total Pages 82

Joint Tactical Radio System (JTRS) Standard Ethernet Device Application Program Interface (API)

Version: 1.2.2
31 March 2008

Statement A- Approved for public release; distribution is unlimited (29 March 2007)

REVISION HISTORY

Version	Authorization	Description	Last Modified Date
1.0		Initial release **ICWG Approved**	23-January-2006
1.1		Update outline format **ICWG Approved**	26-January-2006
1.1.1		Preparation for public release	29-March-2007
1.2 <Draft>		Change Proposals: -Removed Valid Ranges from Provides Services -Added Section B, Mode Configuration Extension -Added Section C, Multicast Mode Extension -Added Section D, Promiscuous Mode Extension -Added Section E, Header Configuration Extension -Added Section F, MAC Address Extension	24-August-2007
1.2		Updates based on ICWG review	01-October-2007
1.2.1		Updates based on ICWG: Added “WF TX and RX directions” clarification in Header Configuration Extension **ICWG Approved**	11-October-2007
1.2.2		Corrected document redlines for public release	31 March 2008

Table of Contents

A. ETHERNET DEVICE 10

B. MODE CONFIGURATION EXTENSION 31

C. MULTICAST MODE EXTENSION 44

D. PROMISCUOUS MODE EXTENSION 51

E. HEADER CONFIGURATION EXTENSION 58

F. MAC ADDRESS EXTENSION 71

Table of Contents

A. ETHERNET DEVICE 10
A.1 Introduction 10
A.1.1 Overview 10
A.1.2 Service Layer Description 11
A.1.2.1 Ethernet Device Port Connections 11
A.1.3 Modes of Service 12
A.1.4 Service States 12
A.1.4.1 Ethernet Device State Diagram 12
A.1.5 Referenced Documents 13
A.1.5.1 Government Documents 13
A.1.5.2 Commercial Standards 13
A.2 Services 14
A.2.1 Provide Services 14
A.2.2 Use Services 16
A.2.3 Interface Modules 18
A.2.3.1 Ethernet 18
A.2.4 Sequence Diagrams 19
A.2.4.1 Transfer Data 19
A.3 Service Primitives and Attributes 23
A.3.1 Ethernet::EthernetPacketConsumer 24
A.3.1.1 *pushPacket* Operation 24
A.4 IDL 25
A.4.1 Ethernet 25
A.5 UML 26
A.5.1 Data Types 27
A.5.2 Enumerations 27
A.5.3 Exceptions 27
A.5.4 Structures 27
Appendix A.A Abbreviations and Acronyms 28
Appendix A.B Performance Specification 29
Appendix A.C Payload Size Specification 30

B. MODE CONFIGURATION EXTENSION 31
B.1 Introduction 31
B.1.1 Overview 31
B.1.2 Service Layer Description 32
B.1.2.1 Mode Configuration Extension Port Connections 32
B.1.3 Modes of Service 33
B.1.4 Service States 33
B.1.5 Referenced Documents 33
B.2 Services 34
B.2.1 Provide Services 34
B.2.2 Use Services 34
B.2.3 Interface Modules 35

B.2.3.1 Ethernet ... 35
B.2.4 Sequence Diagrams ... 35
B.3 Service Primitives and Attributes ... 36
B.3.1 Ethernet::EthernetModeConfig ... 37
B.3.1.1 *getMode* Operation ... 37
B.3.1.2 *setMode* Operation ... 38
B.4 IDL ... 39
B.4.1 EthernetModeConfigExt ... 39
B.5 UML ... 40
B.5.1 Data Types ... 41
B.5.2 Enumerations ... 41
B.5.2.1 Ethernet::EthernetMode ... 41
B.5.3 Exceptions ... 41
B.5.4 Structures ... 41
Appendix B.A Abbreviations and Acronyms ... 42
Appendix B.B Performance Specification ... 43

C. MULTICAST MODE EXTENSION ... 44
C.1 Introduction ... 44
C.1.1 Overview ... 44
C.1.2 Service Layer Description ... 44
C.1.3 Modes of Service ... 44
C.1.4 Service States ... 44
C.1.5 Referenced Documents ... 44
C.2 Services ... 45
C.3 Service Primitives and Attributes ... 45
C.4 IDL ... 46
C.4.1 EthernetMulticastConfigExt ... 46
C.5 UML ... 47
C.5.1 Data Types ... 48
C.5.2 Enumerations ... 48
C.5.3 Exceptions ... 48
C.5.4 Structures ... 48
Appendix C.A Abbreviations and Acronyms ... 49
Appendix C.B Performance Specification ... 50

D. PROMISCUOUS MODE EXTENSION ... 51
D.1 Introduction ... 51
D.1.1 Overview ... 51
D.1.2 Service Layer Description ... 51
D.1.3 Modes of Service ... 51
D.1.4 Service States ... 51
D.1.5 Referenced Documents ... 51
D.2 Services ... 52
D.3 Service Primitives and Attributes ... 52
D.4 IDL ... 53

D.4.1 EthernetPromiscuousConfigExt ... 53
D.5 UML ... 54
D.5.1 Data Types ... 55
D.5.2 Enumerations ... 55
D.5.3 Exceptions ... 55
D.5.4 Structures ... 55
Appendix D.A Abbreviations and Acronyms ... 56
Appendix D.B Performance Specification ... 57

E. HEADER CONFIGURATION EXTENSION ... 58
E.1 Introduction ... 58
E.1.1 Overview ... 58
E.1.2 Service Layer Description ... 59
E.1.2.1 Header Configuration Extension Port Connections ... 59
E.1.3 Modes of Service ... 60
E.1.4 Service States ... 60
E.1.5 Referenced Documents ... 60
E.2 Services ... 61
E.2.1 Provide Services ... 61
E.2.2 Use Services ... 61
E.2.3 Interface Modules ... 62
E.2.3.1 Ethernet ... 62
E.2.4 Sequence Diagrams ... 62
E.3 Service Primitives and Attributes ... 63
E.3.1 Ethernet::EthernetHeaderConfig ... 64
E.3.1.1 *getRetainHeader* Operation ... 64
E.3.1.2 *setRetainHeader* Operation ... 65
E.4 IDL ... 66
E.4.1 EthernetHeaderConfigExt ... 66
E.5 UML ... 67
E.5.1 Data Types ... 68
E.5.2 Enumerations ... 68
E.5.3 Exceptions ... 68
E.5.4 Structures ... 68
Appendix E.A Abbreviations and Acronyms ... 69
Appendix E.B Performance Specification ... 70

F. MAC ADDRESS EXTENSION ... 71
F.1 Introduction ... 71
F.1.1 Overview ... 71
F.1.2 Service Layer Description ... 72
F.1.2.1 MAC Address Extension Port Connections ... 72
F.1.3 Modes of Service ... 73
F.1.4 Service States ... 73
F.1.5 Referenced Documents ... 73
F.2 Services ... 74

F.2.1 Provide Services 74
F.2.2 Use Services 74
F.2.3 Interface Modules 75
F.2.3.1 Ethernet 75
F.2.3.2 Sequence Diagrams 75
F.3 Service Primitives and Attributes 76
F.3.1 Ethernet::EthernetAddress 77
F.3.1.1 *getMacAddress*Operation 77
F.4 IDL 78
F.4.1 EthernetMacAddressExt 78
F.5 UML 79
F.5.1 Data Types 80
F.5.2 Enumerations 80
F.5.3 Exceptions 80
F.5.4 Structures 80
Appendix F.A Abbreviations and Acronyms 81
Appendix F.B Performance Specification 82

Lists of Figures

FIGURE 1 – ETHERNET DEVICE PORT DIAGRAM 11
FIGURE 2 – ETHERNET DEVICE STATE DIAGRAM 12
FIGURE 3 – ETHERNET DEVICE CLASS DIAGRAM 18
FIGURE 4 – DEVICE PROVIDES DATA SEQUENCE DIAGRAM 20
FIGURE 5 – DEVICE CONSUMES DATA SEQUENCE DIAGRAM 22
FIGURE 6 – ETHERNET DEVICE COMPONENT DIAGRAM 26
FIGURE 7 - MODE CONFIGURATION EXTENSION PORT DIAGRAM 32
FIGURE 8 – ETHERNETMODECONFIG INTERFACE DIAGRAM 35
FIGURE 9 – MODE CONFIGURATION EXTENSION COMPONENT DIAGRAM 40
FIGURE 10 – MULTICAST CONFIGURATION EXTENSION COMPONENT DIAGRAM 47
FIGURE 11 – PROMISCUOUS MODE CONFIGURATION EXTENSION COMPONENT DIAGRAM 54
FIGURE 12 - HEADER CONFIGURATION EXTENSION PORT DIAGRAM 59
FIGURE 13 – ETHERNETHEADERCONFIG INTERFACE DIAGRAM 62
FIGURE 14 – HEADER CONFIGURATION EXTENSION COMPONENT DIAGRAM 67
FIGURE 15 - MAC ADDRESS EXTENSION PORT DIAGRAM 72
FIGURE 16 – ETHERNETADDRESS INTERFACE DIAGRAM 75
FIGURE 17 – MAC ADDRESS EXTENSION COMPONENT DIAGRAM 79

List of Tables

TABLE 1 – ETHERNET DEVICE PROVIDE SERVICE INTERFACE 14
TABLE 2 – ETHERNET DEVICE PROVIDE SERVICE AGGREGATE INTERFACE 15
TABLE 3 – ETHERNET DEVICE USE SERVICE INTERFACE 16
TABLE 4 – ETHERNET DEVICE PERFORMANCE SPECIFICATION 29
TABLE 5 – PAYLOAD SIZE PROVIDE SERVICE INTERFACE 30
TABLE 6 – MODE CONFIGURATION EXTENSION PROVIDE SERVICE INTERFACE 34
TABLE 7 – MODE CONFIGURATION EXTENSION PERFORMANCE SPECIFICATION 43
TABLE 8 – HEADER CONFIGURATION EXTENSION PROVIDE SERVICE INTERFACE 61
TABLE 9 – HEADER CONFIGURATION EXTENSION PERFORMANCE SPECIFICATION 70
TABLE 10 – MAC ADDRESS EXTENSION PROVIDE SERVICE INTERFACE 74
TABLE 11 – MAC ADDRESS EXTENSION PERFORMANCE SPECIFICATION 82

A. ETHERNET DEVICE

A.1 INTRODUCTION

The *Ethernet Device* supports methods and attributes that are specific to the Ethernet hardware (HW) device it represents.

This document defines a common set of *Ethernet Device* provide services and interfaces required by most Joint Tactical Radio (JTR) Sets.

The *Ethernet Device* acts as "device adapter". It is used by Common Object Request Broker Architecture (CORBA) components (e.g., waveform application components) to access JTR Set Ethernet HW.

A.1.1 Overview

This base contains as follows:

a. Section A.1, *Introduction*, contains the introductory material regarding the overview, service layer description, modes, states and referenced documents of this document.

b. Section A.2, *Services*, provides summary of service interface uses, interface for each device component, port connections, and sequence diagrams.

c. Section A.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Ethernet Device*.

d. Section A.4, *IDL*.

e. Section A.5, *UML*.

f. Appendix A.A, *Abbreviations and Acronyms.*

g. Appendix A.B, *Performance Specification.*

h. Appendix A.C, *Payload Size Specification.*

A.1.2 Service Layer Description

A.1.2.1 Ethernet Device Port Connections

Figure 1 shows the port connections for the *Ethernet Device* which is an aggregation of the Ethernet and the Device Message Control IDL [1].

Note: All port names are for reference only.

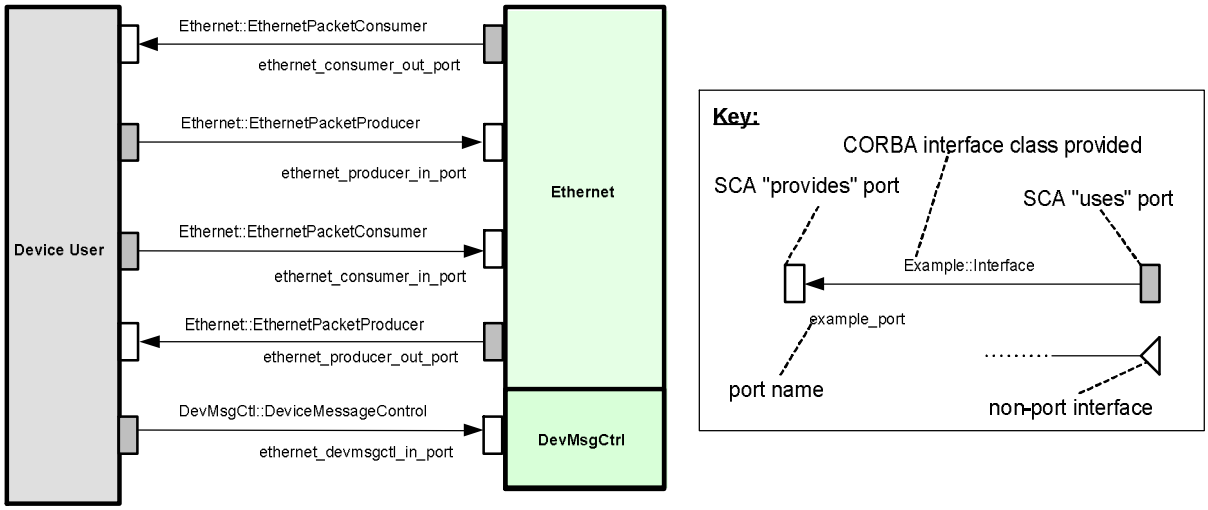

Figure 1 – Ethernet Device Port Diagram

***Ethernet Device* Provides Ports Definitions**

ethernet_consumer_in_port is provided by the *Ethernet Device* to consume packets through operations such as *pushPacket*.
ethernet_producer_in_port is provided by the *Ethernet Device* to provide *DevicePacketSignals* [2] and *DeviceIOSignals* [3] interfaces.
ethernet_devmsgctl_in_port is provided by the *Ethernet Device* to control the data flows.

Ethernet Device Uses Ports Definitions

ethernet_consumer_out_port is used by the *Ethernet Device* to push packet data through operation such as *pushPacket*.
ethernet_producer_out_port is used by the *Ethernet Device* to provide *DevicePacketSignals* [2] and *DeviceIOSignals* [3] interfaces.

***Ethernet Device* Encapsulated Interfaces**

The *Ethernet Device* encapsulates the following CORBA interfaces through aggregate composition. Detail definition of these interfaces and services is provided by separate JTRS API documentation.

- Device Message Control [1]

A.1.3 Modes of Service

Not applicable.

A.1.4 Service States

A.1.4.1 Ethernet Device State Diagram

The *Ethernet Device* states are illustrated in the following diagram. The *Ethernet Device* states ensure that received operations are only executed when the *Ethernet Device* is in the proper state. The five states of the *Ethernet Device* are as follow:

- CONSTRUCTED - The state transitioned to upon successful creation.
- INITIALIZED - The state transitioned to upon successful initialization.
- ENABLED - The state transitioned to upon successful start.
- DISABLED - The state transitioned to upon successful stop.
- RELEASED - The state transitioned to upon successful release.

The *Ethernet Device* transitions between states in response to the *initialize*, *start*, *stop* and *releaseObject* operations [6].

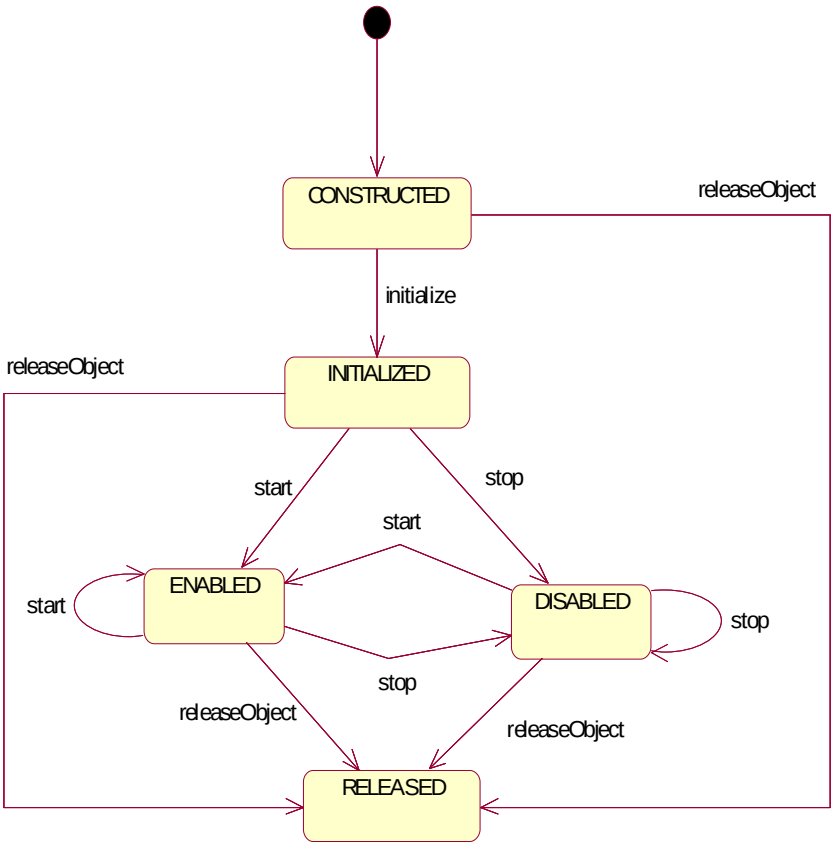

Figure 2 – Ethernet Device State Diagram

A.1.5 Referenced Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein.

A.1.5.1 Government Documents

The following documents are part of this specification as specified herein.

A.1.5.1.1 Specifications

A.1.5.1.1.1 Federal Specifications

None

A.1.5.1.1.2 Military Specifications

None

A.1.5.1.2 Other Government Agency Documents

[1] JTRS Standard, “Device Message Control API,” JPEO, Version 1.1.2.
[2] JTRS Standard, “Device Packet Signals API,” JPEO, Version 1.2.2.
[3] JTRS Standard, “Device IO Signals API,” JPEO, Version 1.1.1.
[4] JTRS Standard, “Device Packet API,” JPEO, Version 1.1.1.
[5] JTRS Standard, “Device IO Control API,” JPEO, Version 1.1.1.
[6] JTRS Standard, “Software Communications Architecture (SCA),” JPEO, Version 2.2.2.
[7] JTRS Standard, “JTRS CORBA Types,” JPEO, Version 1.0.2.

A.1.5.2 Commercial Standards

None

A.2 SERVICES

A.2.1 Provide Services

The *Ethernet Device* provides service consists of the Table 1 service ports, interfaces, and primitives, which can be called by other client components. Detail definition of the interfaces and services shaded in grey is provided by separate JTRS API documentation referenced in the table.

Table 1 – Ethernet Device Provide Service Interface

Service Group (Port Name)	Service (Interface Provided)		Primitives (Provided)	Parameter Name or Return Value	Valid Range
ethernet_consumer _in_port	Ethernet:: EthernetPacket Consumer		pushPacket()	See section A.3 Service Primitives and Attributes	See section A.3 Service Primitives and Attributes
		DevPK::DevicePacket [4]	getMaxPayloadSize()	*Return Value*	See Appendix A.C
			getMinPayloadSize()	*Return Value*	See Appendix A.C
			getDesiredPayloadSize()	*Return Value*	See Appendix A.C
			getMinOverrideTimeout()	*Return Value*	See Appendix A.C
			getNumOfPriorities()	*Return Value*	1
			setNumOfPriorities()	numOfPriorities	1
			enableFlowResumeSignals()	enable	TRUE or FALSE
			enableEmptySignals()	enable	TRUE or FALSE
			spaceAvailable()	priorityQueueID	1
				Return Value	TRUE or FALSE

Service Group (Port Name)	Service (Interface Provided)		Primitives (Provided)	Parameter Name or Return Value	Valid Range
		DevIOC:: DeviceIOControl [5]	enableRTSCTS()	enable	TRUE or FALSE
			setRTS()	RTS	TRUE or FALSE
ethernet_producer_in_port	Ethernet:: EthernetPacket Producer	DevPktSig:: DevicePacketSignals [2]	setMaxPayloadSize()	maxPayloadSize	See Appendix A.C
			setMinPayloadSize()	minPayloadSize	See Appendix A.C
			setDesiredPayloadSize()	desiredPayloadSize	See Appendix A.C
			setMinOverrideTimeout()	minOverrideTimeout	See Appendix A.C
			signalEmpty()	streamId	0 – 65535
			signalFlowResume()	priorityQueueID	1
		DevIOS:: DeviceIOSignals [3]	setCTS()	CTS	TRUE or FALSE

The *Ethernet Device* provides service aggregates the Table 2 service ports, interfaces, and primitives, which can be called by other client components. Detail definition of the interfaces and services shaded in grey is provided by separate JTRS API documentation referenced in the table.

Table 2 – Ethernet Device Provide Service Aggregate Interface

Service Group (Port Name)	Service (Interface Provided)	Primitives (Provided)	Parameter Name or Return Value	Valid Range
ethernet_devmsgctl_in_port	DevMsgCtl::DeviceMessageControl [1]	rxActive()	*Return Value*	TRUE or FALSE
		txActive()	*Return Value*	TRUE or FALSE
		abortTx	streamId	0 - 65535

A.2.2 Use Services

The *Ethernet Device* use service set consists of the Table 3 service ports, interfaces, and primitives. Since the *Ethernet Device* acts as a client with respect to these services from other components, it is required to connect these ports with corresponding service ports applied by the server component. The *Ethernet Device* uses the Port Name as connectionID for the connection. Detail definition of the interfaces and services shaded in grey is provided by separate JTRS API documentation referenced in the table.

Table 3 – Ethernet Device Use Service Interface

Service Group (Port Name)	Service (Interface Used)		Primitives (Used)	Parameter Name or Return Value	Valid Range
ethernet_consumer_out_port	Ethernet:: EthernetPacket Consumer		pushPacket()	See section A.3 Service Primitives and Attributes	See section A.3 Service Primitives and Attributes
		DevPK::DevicePacket [4]	getMaxPayloadSize()	*Return Value*	512 to 16383
			getMinPayloadSize()	*Return Value*	0 to 512
			getDesiredPayloadSize()	*Return Value*	512 to 16383
			getMinOverrideTimeout()	*Return Value*	0 to 50
			getNumOfPriorities()	*Return Value*	1
			setNumOfPriorities()	numOfPriorities	1
			enableFlowResumeSignals()	enable	TRUE or FALSE
			enableEmptySignals()	enable	TRUE or FALSE
			spaceAvailable()	priorityQueueID	1
				Return Value	TRUE or FALSE
		DevIOC:: DeviceIOControl [5]	enableRTSCTS()	enable	TRUE or FALSE
			setRTS()	RTS	TRUE or FALSE

Service Group (Port Name)	Service (Interface Used)		Primitives (Used)	Parameter Name or Return Value	Valid Range
ethernet_producer_ out_port	Ethernet:: EthernetPacket Producer	DevPktSig:: DevicePacketSignals [2]	setMaxPayloadSize()	maxPayloadSize	1 to 16383
			setMinPayloadSize()	minPayloadSize	0 to 512
			setDesiredPayloadSize()	desiredPayloadSize	512 to 16383
			setMinOverrideTimeout()	minOverrideTimeout	0 to 50
			signalEmpty()	*None*	N/A
			signalFlowResume()	priorityQueueID	1
		DevIOS:: DeviceIOSignals [3]	setCTS()	CTS	TRUE or FALSE

A.2.3 Interface Modules

A.2.3.1 Ethernet

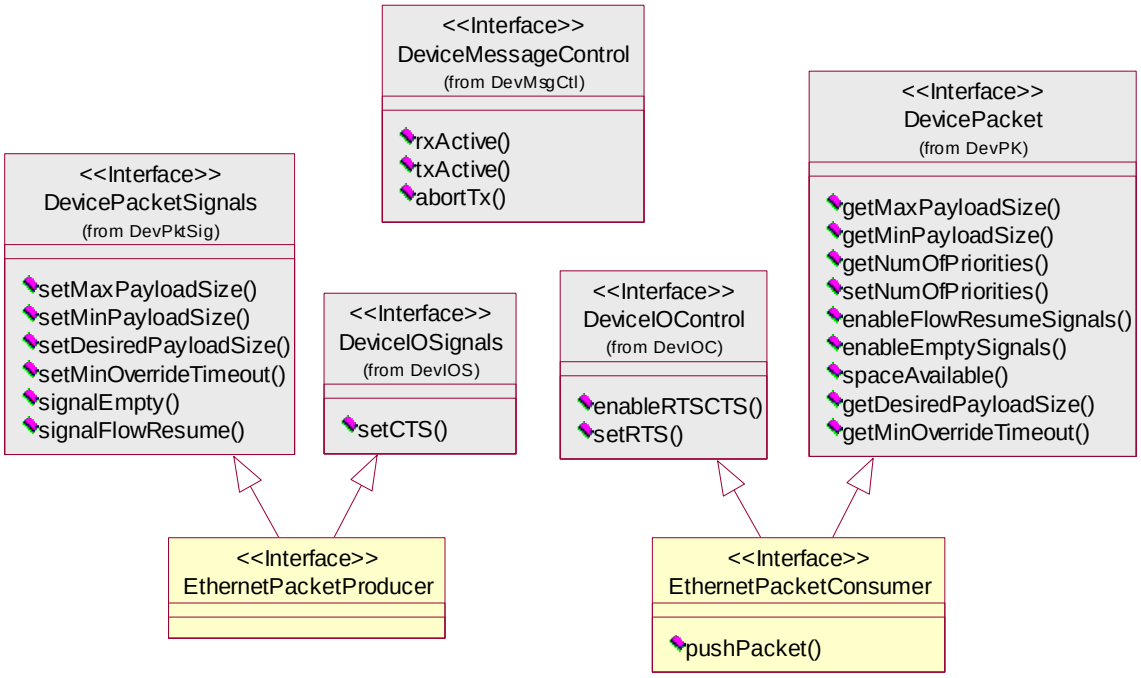

Figure 3 – Ethernet Device Class Diagram

A.2.3.1.1 EthernetPacketConsumer Interface Description

The interface design of the *EthernetPacketConsumer* is shown in Figure 3. It extends the *DeviceIOControl* [5] and *DevicePacket* [4] interfaces to provide the ability to transfer data packets to Ethernet HW with flow control.

A.2.3.1.2 EthernetPacketProducer Interface Description

The interface design of the *EthernetPacketProducer* is shown in Figure 3. It extends the *DeviceIOSignals* [3] and *DevicePacketSignals* [2] interfaces to provide the ability to transfer data packets from Ethernet HW with flow control.

A.2.4 Sequence Diagrams

A.2.4.1 Transfer Data

The Transfer Device Use Case covers scenarios in which data is transferred to or from a Radio Device. The scenarios are dependent on whether the *Device* needs to provide or consume data.

A.2.4.1.1 Device Provides Data Sequence

Description

A *Device* provides data as shown in Figure 4. The *Device* in this sequence is the *Ethernet Device*. The *Device User* in the sequence is the *EthernetPacketConsumer*.

Upon connection to the *Device User*, the *Device* issues *enableRTSCTS* and *enableFlowResumeSignal* commands to the *Device User*. When data is available, the *Device* pushes it to the *Device User* using the *pushPacket* method. In this diagram, the flow control mechanism that can be used by the *Device User* to halt *pushPacket*(s) is demonstrated when the *pushPacket* returns FALSE. Ordinarily this is issued if the *Device User* cannot hold at least one more packet of *maxPayloadSize*. Once the *Device User* is ready to receive more packets, it issues a *signalFlowResume* to the *Device*. Once this is received, the *Device* resumes pushing data to the *Device User* using the *pushPacket* method.

Pre-conditions

The *Device* is in the ENABLED state.

Post-conditions

The *Device* is pushing incoming data to the *Device User* and responding to flow control as needed.

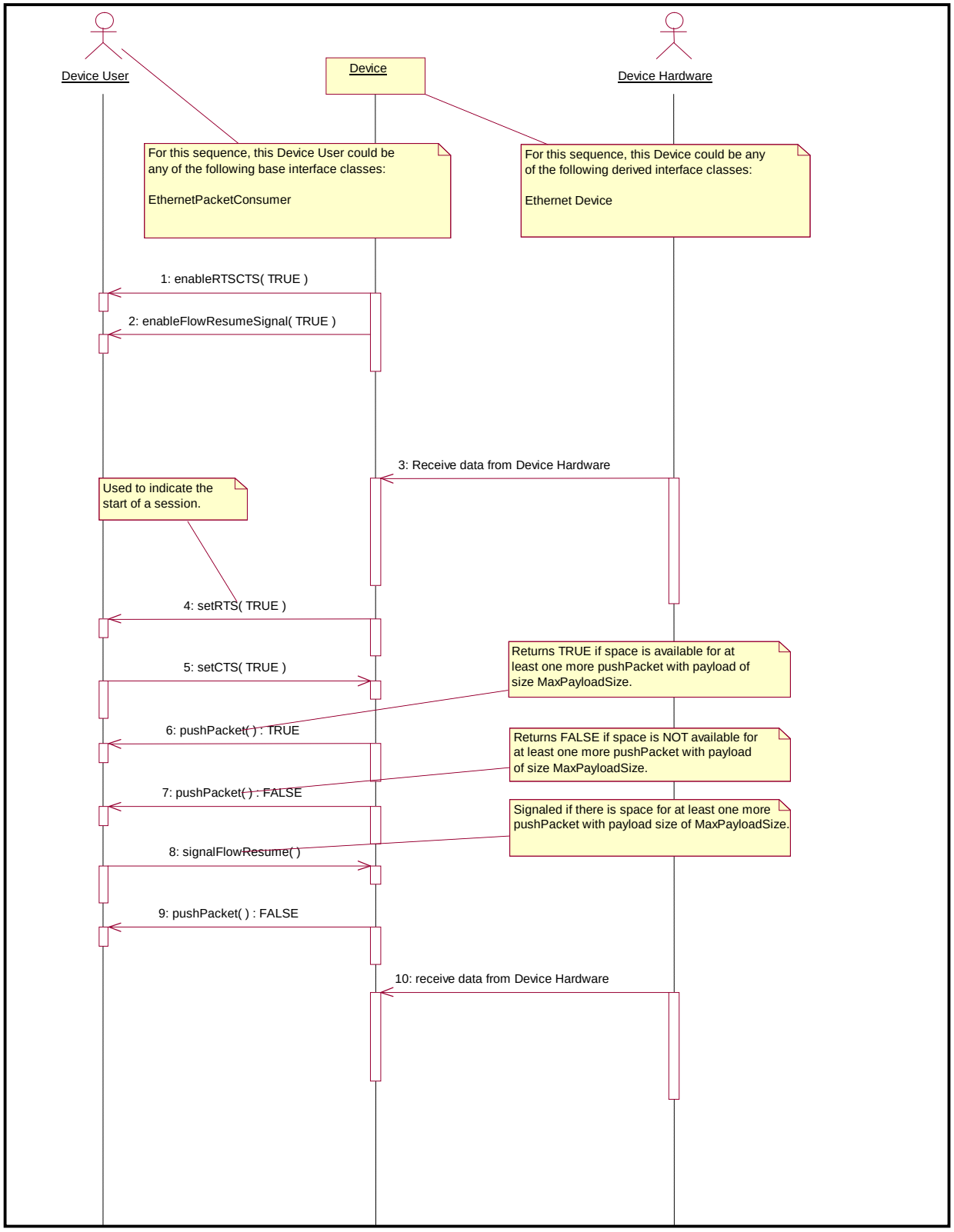

Figure 4 – Device Provides Data Sequence Diagram

A.2.4.1.2 Device Consumes Data Sequence

Description

A *Device* consumes data as shown in Figure 5. The *Device* in this sequence is the *Ethernet Device.* The *Device User* in the sequence derives the *DevicePacketSignals* and *DeviceIOSignals* base classes.

Upon connection to the *Device*, the *Device User* issues *enableRTSCTS* and *enableFlowResumeSignal* commands by the *Device*. Data to be sent to the Device HW is transferred to the *Device* through the *pushPacket* method. In this diagram, the flow control mechanism that can be used by the *Device* to halt *pushPacket*(s) is demonstrated when the *pushPacket* returns FALSE. This is issued if the *Device* cannot hold one more packet of *maxPayloadSize*. Once the *Device* is ready to receive more packets, a *signalFlowResume* signal is issued to the *Device User*. Once this is received, the *Device User* resumes pushing data to the *Device* using the *pushPacket* method.

Pre-conditions

The *Device* is in the ENABLED state.

Post-conditions

The *Device User* is pushing outgoing data to the *Device* and responding to flow control signals as needed.

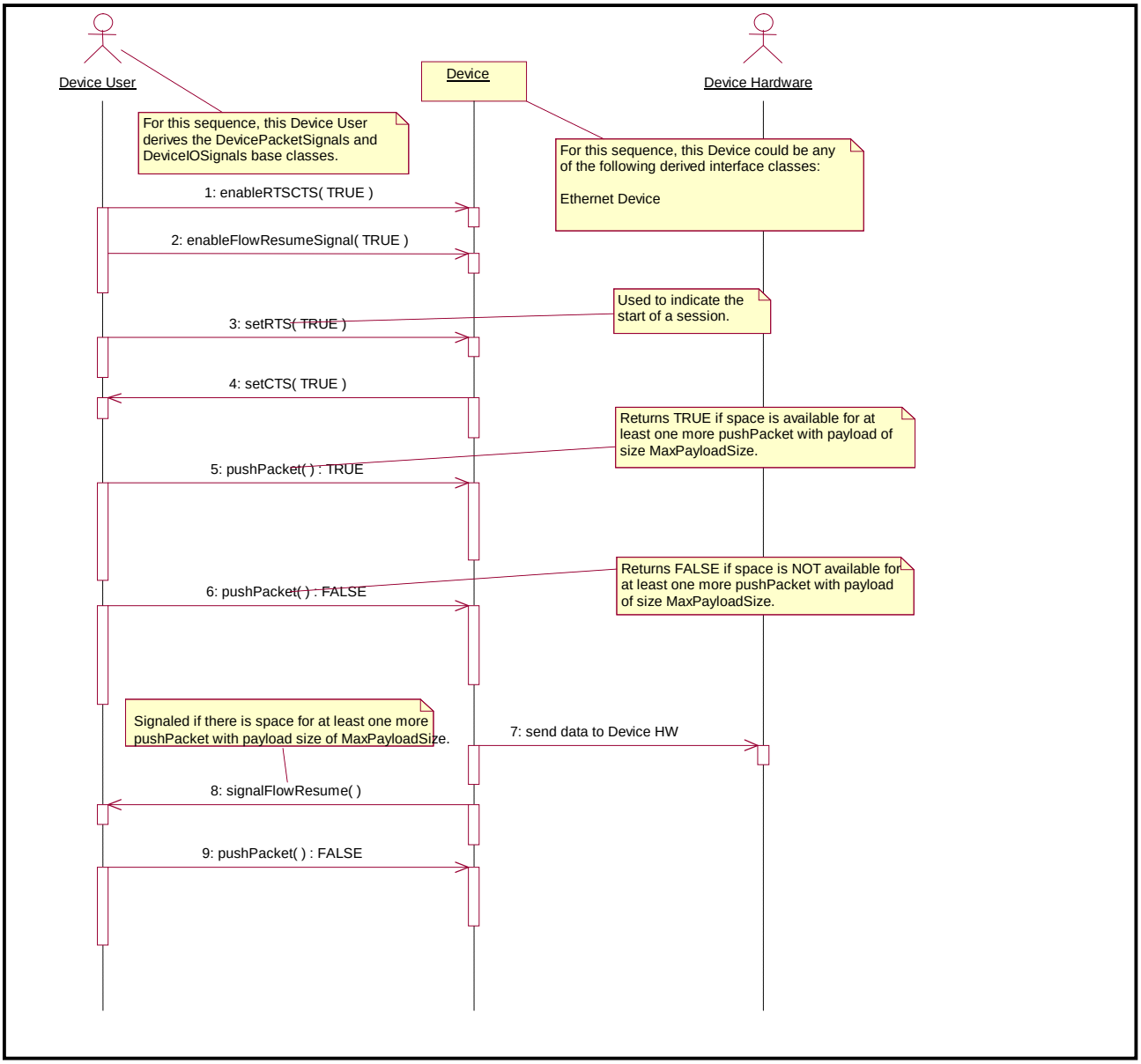

Figure 5 – Device Consumes Data Sequence Diagram

A.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section A.5. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

A.3.1 Ethernet::EthernetPacketConsumer

A.3.1.1 *pushPacket* Operation

The *pushPacket* operation provides the ability to push Ethernet Port data packets to the packet consumer.

A.3.1.1.1 Synopsis

boolean pushPacket(in octet priority, in CF::OctetSequence address, in CF::OctetSequence payload);

A.3.1.1.2 Parameters

Parameter Name	Description	Type	Units	Valid Range
priority	Indicates the priority queue to put the address and associated payload in.	octet	Priority Level	1
address	A sequence of octet containing the Media Access Control (MAC) address.	CF::OctetSequence (See *SCA* [6])	byte	N/A
payload	A sequence of octet containing real-time data to be pushed.	CF::OctetSequence (See *SCA* [6])	byte	N/A

A.3.1.1.3 State

ENABLED CF::Device::operationalState.

A.3.1.1.4 New State

This operation does not cause a state change.

A.3.1.1.5 Return Value

Description	Type	Units	Valid Range
Whether the data can be buffered to the packet consumer or not.	boolean	N/A	TRUE = data can be pushed to the packet consumer's buffer. FALSE = The packet consumer's buffer is full, data cannot be pushed.

A.3.1.1.6 Originator

Service User

A.3.1.1.7 Exceptions

None

A.4 IDL

A.4.1 Ethernet

```
/*
** Ethernet.idl
**/

#ifndef __ETHERNET_DEFINED
#define __ETHERNET_DEFINED

#ifndef __DEVICEIOCONTROL_DEFINED
#include "DeviceIoControl.idl"
#endif

#ifndef __DEVICEIOSIGNALS_DEFINED
#include "DeviceIoSignals.idl"
#endif

#ifndef __DEVICEPACKET_DEFINED
#include "DevicePacket.idl"
#endif

#ifndef __DEVICEPACKETSIGNALS_DEFINED
#include "DevicePacketSignals.idl"
#endif

#ifndef __CF_DEFINED
#include "CF.idl"
#endif

module Ethernet {

	interface EthernetPacketConsumer :	DevIOC::DeviceIOControl,
						DevPK::DevicePacket {
		boolean pushPacket (
			in octet priority,
			in CF::OctetSequence address,
			in CF::OctetSequence payload
			);

	};

	interface EthernetPacketProducer :	DevIOS::DeviceIOSignals,
						DevPktSig::DevicePacketSignals {
	};

};

#endif
```

A.5 UML

This section contains Figure 6 and the definitions of all data types referenced (directly or indirectly) by section A.3 Service Primitives and Attributes.

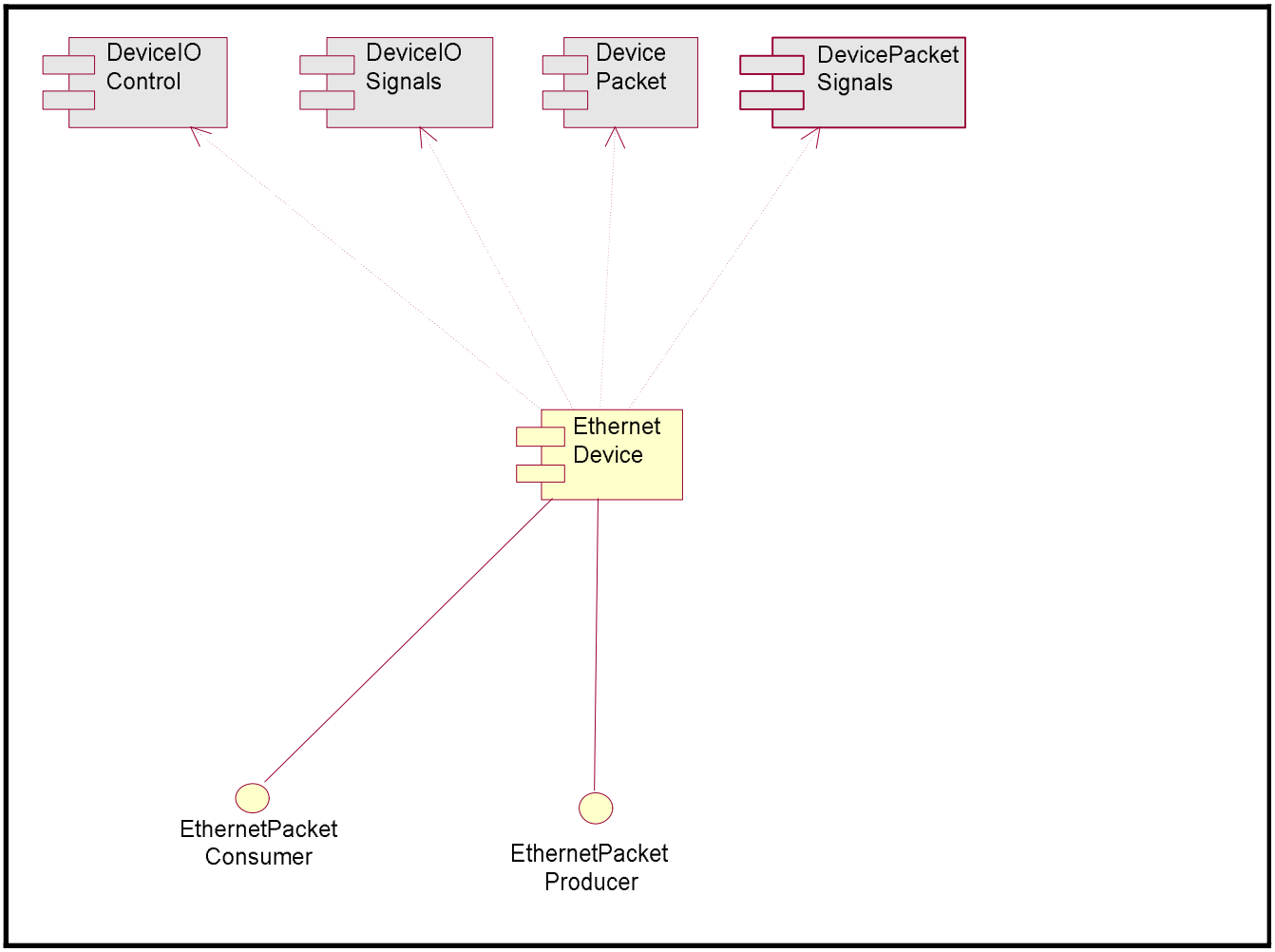

Figure 6 – Ethernet Device Component Diagram

A.5.1 Data Types

None

A.5.2 Enumerations

None

A.5.3 Exceptions

None

A.5.4 Structures

None

APPENDIX A.A ABBREVIATIONS AND ACRONYMS

API	Application Program Interface
CF	Core Framework
CORBA	Common Object Request Broker Architecture
CTS	Clear to Send
HW	Hardware
ICWG	Interface Control Working Group
IDL	Interface Definition Language
JPEO	Joint Program Executive Office
JTRS	Joint Tactical Radio System
MAC	Media Access Control
N/A	Not Applicable
ORB	Object Request Broker
RTS	Request to Send
SCA	Software Communications Architecture
UML	Unified Modeling Language

APPENDIX A.B PERFORMANCE SPECIFICATION

The following table provides a template for the generic performance specification for the *Ethernet Device API*. This performance specification corresponds to the port diagram in Figure 1.

Table 4 – Ethernet Device Performance Specification

Specification	Description	Units	Value
Worst Case Command Execution Time for pushPacket() on ethernet_consumer_in_port	*	*	*
Worst Case Command Execution Time for ethernet_consumer_in_port	*	*	*
Worst Case Command Execution Time for ethernet_producer_in_port	*	*	*
Worst Case Command Execution Time for ethernet_devmsgctl_in_port	*	*	*
Worst Case Execution Time for pushPacket() on ethernet_consumer_out_port	*	*	*
Worst Case Command Execution Time for ethernet_consumer_out_port	*	*	*
Worst Case Command Execution Time for ethernet_producer_out_port	*	*	*

*Note this template will be filled in by individual developers.

APPENDIX A.C PAYLOAD SIZE SPECIFICATION

Table 5 specifies a template for the valid ranges for the *EthernetDevice* payload size parameters or return values.

Table 5 – Payload Size Provide Service Interface

Service Group (Port Name)	Service (Interface Provided)		Primitives (Provided)	Parameter Name or Return Value	Valid Range
ethernet_consumer_in_port	Ethernet:: EthernetPacket Consumer	DevPK::DevicePacket [4]	getMaxPayloadSize()	*Return Value*	*
			getMinPayloadSize()	*Return Value*	*
			getDesiredPayloadSize()	*Return Value*	*
			getMinOverrideTimeout()	*Return Value*	*
ethernet_producer_in_port	Ethernet:: EthernetPacket Producer	DevPktSig:: DevicePacketSignals [2]	setMaxPayloadSize()	maxPayloadSize	*
			setMinPayloadSize()	minPayloadSize	*
			setDesiredPayloadSize()	desiredPayloadSize	*
			setMinOverrideTimeout()	minOverrideTimeout	*

Note: (*) These values should be filled in by individual developers.

B. MODE CONFIGURATION EXTENSION

B.1 INTRODUCTION

The *Mode Configuration Extension* is based upon the *Ethernet Device API*. It extends the functionality of the *Ethernet Device* to set and get the ethernet configuration mode.

B.1.1 Overview

This extension contains as follows:

a. Section B.1, *Introduction*, contains the introductory material regarding the overview, service layer description, modes, states and referenced documents of this document.

b. Section B.2, *Services,* provides summary of service interface uses, interface for each device component, port connections, and sequence diagrams.

c. Section B.3, *Service Primitives and Attributes,* specifies the operations that are provided by the *Ethernet Device* Mode Configuration Extension.

d. Section B.4, *IDL*.

e. Section B.5, *UML*.

f. Appendix B.A, *Abbreviations and Acronyms*

g. Appendix B.B, *Performance Specification*

B.1.2 Service Layer Description

B.1.2.1 Mode Configuration Extension Port Connections

Figure 7 shows the port connections for *Mode Configuration Extension.*

Note: All port names are for reference only. Ports identified in black are provided in A. Ethernet Device.

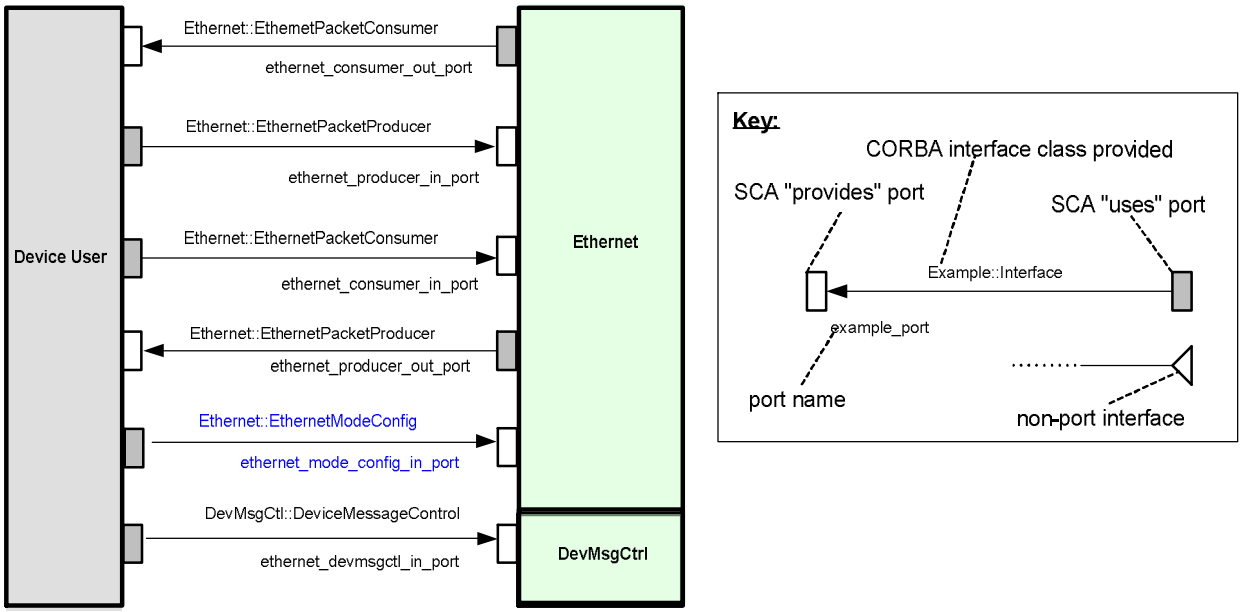

Figure 7 - Mode Configuration Extension Port Diagram

***Mode Configuration Extension* Provides Ports Definitions**.

ethernet_mode_config_in_port is provided by the *Ethernet Device* to provide the ability to set and get the ethernet configuration mode.

***Mode Configuration Extension* Uses Ports Definitions**

None

B.1.3 Modes of Service

Not applicable.

B.1.4 Service States

There are no changes from A. Ethernet Device.

B.1.5 Referenced Documents

There are no changes from A. Ethernet Device

B.2 SERVICES

B.2.1 Provide Services

The *Mode Configuration Extension* provides service consists of the following service ports, interfaces, and primitives, which can be called by other client components.

Table 6 – Mode Configuration Extension Provide Service Interface

Service Group (Port Name)	**Service (Interface Provided)**	**Primitives (Provided)**
ethernet_mode_config_in_port	Ethernet::EthernetMode	getMode()
		setMode()

B.2.2 Use Services

None

B.2.3 Interface Modules

B.2.3.1 Ethernet

B.2.3.1.1 EthernetModeConfig Interface Description

The *Mode Configuration Extension* is shown in the diagram below. The *EthernetModeConfig* interface provides the ability to set and get the ethernet configuration mode.

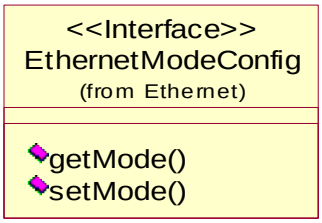

Figure 8 – EthernetModeConfig Interface Diagram

B.2.4 Sequence Diagrams

None

B.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section B.5, UML. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

B.3.1 Ethernet::EthernetModeConfig

B.3.1.1 *getMode* Operation

The *getMode* operation allows the user determine what mode the device is currently configured for.

B.3.1.1.1 Synopsis

```
EthernetMode getMode();
```

B.3.1.1.2 Parameters

None

B.3.1.1.3 State

ENABLED CF::Device::operationalState.

B.3.1.1.4 New State

This operation does not cause a state change.

B.3.1.1.5 Return Value

Type	Description
EthernetMode (See Section B.5.2.1)	The current mode the *Device* is configured for.

B.3.1.1.6 Originator

Service User

B.3.1.1.7 Exceptions

None

B.3.1.2 *setMode* Operation

The *setMode* operation allows the user to set the device to a particular mode.

B.3.1.2.1 Synopsis

void setMode(in EthernetMode mode) raises(JTRS::Unsupported);

B.3.1.2.2 Parameters

Parameter Name	Type	Description
mode	EthernetMode (See Section B.5.2.1)	The mode to configure the Device to.

B.3.1.2.3 State

ENABLED CF::Device::operationalState.

B.3.1.2.4 New State

This operation does not cause a state change.

B.3.1.2.5 Return Value

None

B.3.1.2.6 Originator

Service User

B.3.1.2.7 Exceptions

Type	Description
JTRS::Unsupported (See *JTRS CORBA Types* [7])	Indicates the requested mode is not supported by this platform

B.4 IDL

B.4.1 EthernetModeConfigExt

```
/*
** EthernetModeConfigExt.idl
*/

#ifndef __ETHERNETMODECONFIGEXT_DEFINED
#define __ETHERNETMODECONFIGEXT_DEFINED

#ifndef __JTRSCORBATYPES_DEFINED
#include "JtrsCorbaTypes.idl"
#endif

#ifndef __ETHERNET_DEFINED
#include "Ethernet.idl"
#endif

module Ethernet {

	typedef JTRS::ExtEnum  EthernetMode;

	const EthernetMode MODE_NONE = 0;

	interface EthernetModeConfig {

		EthernetMode getMode();

		void setMode( in EthernetMode mode )  raises( JTRS::Unsupported );
	};

	const EthernetMode MODE_STANDARD = MODE_NONE + 1;

};

#endif // __ETHERNETMODECONFIGEXT_DEFINED
```

B.5 UML

This section contains the Device component UML diagram and the definitions of all data types referenced (directly or indirectly) by the Service Primitives and Attributes in B.3.

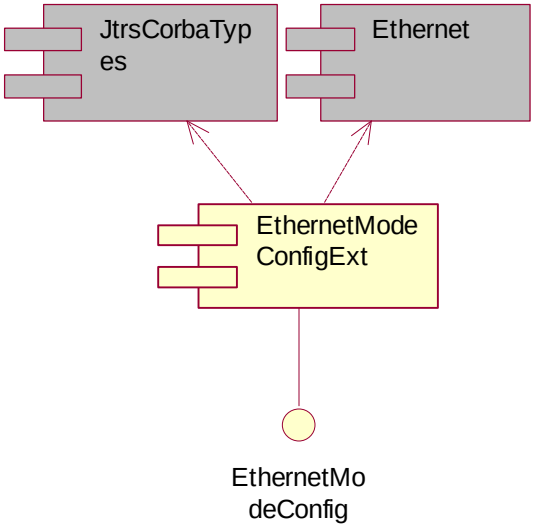

Figure 9 – Mode Configuration Extension Component Diagram

B.5.1 Data Types

None

B.5.2 Enumerations

B.5.2.1 Ethernet::EthernetMode

The *EthernetMode* type definition is a JTRS extension enumeration (see *JTRS CORBA Types* [7]). Additional modes supported by the *Ethernet Device* will be defined in their respective extensions.

```
typedef JTRS::ExtEnum  EthernetMode;
const EthernetMode MODE_NONE = 0;
const EthernetMode MODE_STANDARD = MODE_NONE + 1;
```

JTRS::ExtEnum	Element	Value	Description
EthernetMode	MODE_NONE	0	No Mode
	MODE_STANDARD	MODE_NONE + 1	Standard Mode

B.5.3 Exceptions

None

B.5.4 Structures

None

APPENDIX B.A ABBREVIATIONS AND ACRONYMS

There are no changes from A. Ethernet Device.

APPENDIX B.B PERFORMANCE SPECIFICATION

Table 7 provides a template for the generic performance specification for the *Ethernet Device Mode Configuration Extension* which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 7.

Table 7 – Mode Configuration Extension Performance Specification

Specification	Description	Units	Value
Worst Case Command Execution Time for ethernet_mode_config_in_port	*	*	*

Note: (*) These values should be filled in by individual developers.

C. MULTICAST MODE EXTENSION

C.1 INTRODUCTION

The *Multicast Mode Extension* is based upon the *Ethernet Device Mode Configuration Extension* (see section B). It extends the functionality of the *Ethernet Device* to include multicast mode capabilities.

C.1.1 Overview

This extension contains as follows:

a. Section C.1, *Introduction*, contains the introductory material regarding the overview, service layer description, modes, states, and referenced documents of this document.

b. Section C.2, *Services*, provides summary of service interface uses, interface for each device component, port connections, and sequence diagrams.

c. Section C.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Ethernet Device* Multicast Mode Extension.

d. Section C.4, *IDL*.

e. Section C.5, *UML*.

f. Appendix C.A, *Abbreviations and Acronyms*.

g. Appendix C.B, *Performance Specification*.

C.1.2 Service Layer Description

There are no changes from A. Ethernet Device.

C.1.3 Modes of Service

Not applicable.

C.1.4 Service States

There are no changes from A. Ethernet Device.

C.1.5 Referenced Documents

There are no changes from A. Ethernet Device.

C.2 SERVICES

There are no changes from A. Ethernet Device.

C.3 SERVICE PRIMITIVES AND ATTRIBUTES

There are no changes from A. Ethernet Device.

C.4 IDL

C.4.1 EthernetMulticastConfigExt

```
/*
** EthernetMulticastConfigExt.idl
*/

#ifndef __ETHERNETMULTICASTCONFIGEXT_DEFINED
#define __ETHERNETMULTICASTCONFIGEXT_DEFINED

#ifndef __ETHERNETMODECONFIGEXT_DEFINED
#include "EthernetModeConfigExt.idl"
#endif

module Ethernet {

	const EthernetMode MODE_MULTICAST = MODE_NONE + 2;

};

#endif // __ETHERNETMULTICASTCONFIGEXT_DEFINED
```

C.5 UML

This section contains the UML diagram and the definitions of all data types referenced (directly or indirectly) by the Service Primitives and Attributes in section C.3.

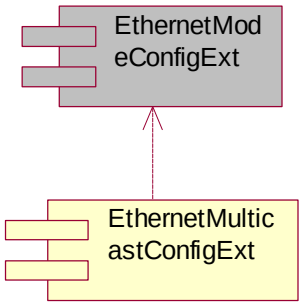

Figure 10 – `Multicast Configuration Extension Component Diagram`

C.5.1 Data Types

None

C.5.2 Enumerations

C.5.2.1 Ethernet::EthernetMode

The following specifies the additional modes supported by the *Ethernet Device Mode Configuration Extension* (see section B) that is defined in this extension.

```
const EthernetMode MODE_MULTICAST = MODE_NONE + 2;
```

JTRS::ExtEnum	Element	Value	Description
EthernetMode (See Section B.5.2.1)	MODE_MULTICAST	MODE_NONE + 2	Multicast Mode

C.5.3 Exceptions

None

C.5.4 Structures

None

APPENDIX C.A ABBREVIATIONS AND ACRONYMS

There are no changes from A. Ethernet Device.

APPENDIX C.B PERFORMANCE SPECIFICATION

There are no changes from A. Ethernet Device.

D. PROMISCUOUS MODE EXTENSION

D.1 INTRODUCTION

The *Promiscuous Mode Extension* is based upon the *Ethernet Device Mode Configuration Extension* (see section B). It extends the functionality of the *Ethernet Device* to include promiscuous mode capabilities.

D.1.1 Overview

This extension contains as follows:

a. Section D.1, *Introduction*, contains the introductory material regarding the overview, service layer description, modes, states, and referenced documents of this document.

b. Section D.2, *Services*, provides summary of service interface uses, interface for each device component, port connections, and sequence diagrams.

c. Section D.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Ethernet Device* Promiscuous Mode Extension.

d. Section D.4, *IDL*.

e. Section D.5, *UML*.

f. Appendix D.A, *Abbreviations and Acronyms.*

g. Appendix D.B, *Performance Specification.*

D.1.2 Service Layer Description

There are no changes from A. Ethernet Device.

D.1.3 Modes of Service

Not applicable.

D.1.4 Service States

There are no changes from A. Ethernet Device.

D.1.5 Referenced Documents

There are no changes from A. Ethernet Device.

D.2 SERVICES

There are no changes from A. Ethernet Device.

D.3 SERVICE PRIMITIVES AND ATTRIBUTES

There are no changes from A. Ethernet Device.

D.4 IDL

D.4.1 EthernetPromiscuousConfigExt

```
/*
** EthernetPromiscuousConfigExt.idl
*/

#ifndef __ETHERNETPROMISCUOUSCONFIGEXT_DEFINED
#define __ETHERNETPROMISCUOUSCONFIGEXT_DEFINED

#ifndef __ETHERNETMODECONFIGEXT_DEFINED
#include "EthernetModeConfigExt.idl"
#endif

module Ethernet {

	const EthernetMode MODE_PROMISCUOUS = MODE_NONE + 3;

};

#endif // __ETHERNETPROMISCUOUSCONFIGEXT_DEFINED
```

D.5 UML

This section contains the UML diagram and the definitions of all data types referenced (directly or indirectly) by the Service Primitives and Attributes in section D.3.

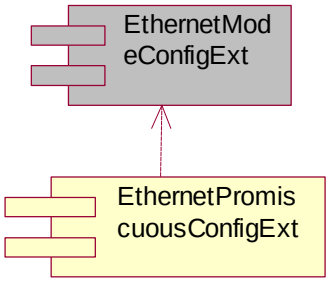

Figure 11 – Promiscuous Mode Configuration Extension Component Diagram

D.5.1 Data Types

None

D.5.2 Enumerations

D.5.2.1 Ethernet::EthernetMode

The following specifies the additional modes supported by the *Ethernet Device Mode Configuration Extension* (see section B) that is defined in this extension.

```
const EthernetMode MODE_PROMISCUOUS = MODE_NONE + 3;
```

JTRS::ExtEnum	Element	Value	Description
EthernetMode (See Section B.5.2.1	MODE_PROMISCUOUS	MODE_NONE + 3	Promiscuous Mode

D.5.3 Exceptions

None

D.5.4 Structures

None

APPENDIX D.A ABBREVIATIONS AND ACRONYMS

There are no changes from A. Ethernet Device.

APPENDIX D.B PERFORMANCE SPECIFICATION

There are no changes from A. Ethernet Device.

E. HEADER CONFIGURATION EXTENSION

E.1 INTRODUCTION

The *Header Configuration Extension* is based upon the *Ethernet Device API*. It extends the functionality of the *Ethernet Device* to allow a user to retain/accept the ethernet Media Access Control (MAC) header on packets being pushed to/from the *Device Use*.

The *Header Configuration Extension* applies to both WF TX and RX directions. When enabled, in the WF TX direction the packet passed to the WF in the *pushPacket* "payload" parameter will have the MAC header retained and untouched. When enabled, in the WF RX direction the packet passed to the *Device* in the *pushPacket* "payload" parameter should have the complete MAC header already attached as the Device will simply pass this packet to the Ethernet HW "as is". When enabled, in either direction the "address" parameter of the *pushPacket* should be set to 0 length by the client and ignored servant.

E.1.1 Overview

This extension contains as follows:

a. Section E.1, *Introduction*, contains the introductory material regarding the overview, service layer description, modes, states and referenced documents of this document.

b. Section E.2, *Services*, provides summary of service interface uses, interface for each device component, port connections, and sequence diagrams.

c. Section E.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Ethernet Device* Header Configuration Extension.

d. Section E.4, *IDL*.

e. Section E.5, *UML*.

f. Appendix E.A, *Abbreviations and Acronyms*

g. Appendix E.B, *Performance Specification*

E.1.2 Service Layer Description

E.1.2.1 Header Configuration Extension Port Connections

Figure 12 shows the port connections for *Header Configuration Extension.*

Note: All port names are for reference only. Ports identified in black are provided in A. Ethernet Device.

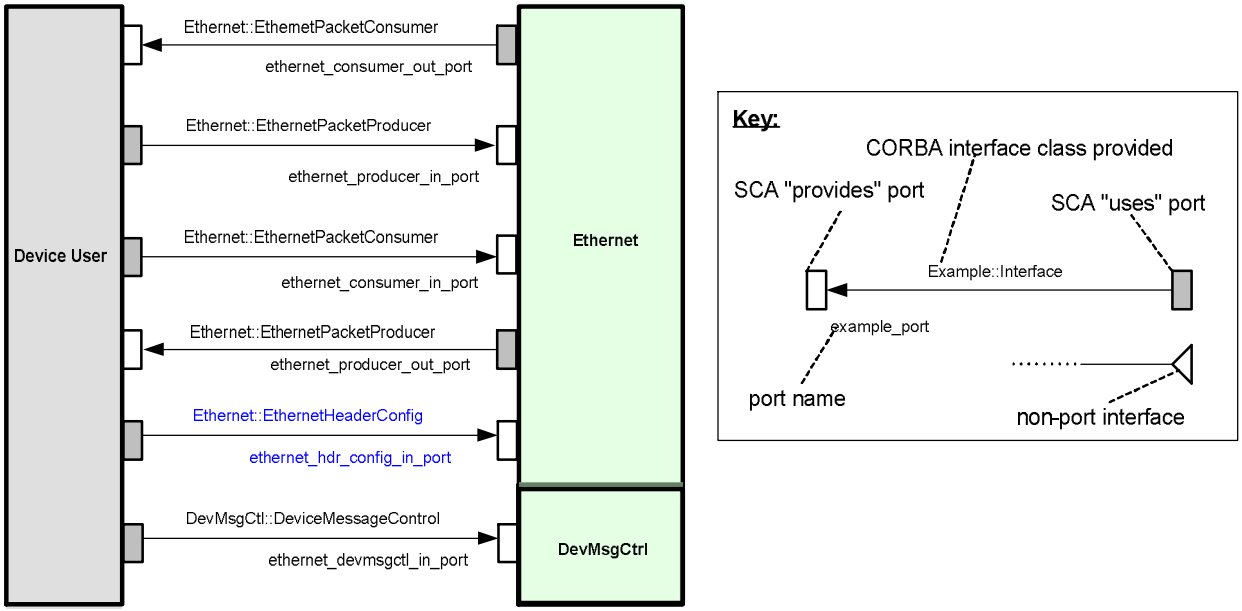

Figure 12 - Header Configuration Extension Port Diagram

***Mode Configuration Extension* Provides Ports Definitions**.

ethernet_hdr_config_in_port is provided by the *Ethernet Device* to retain/accept the ethernet MAC header on packets being pushed to/from the *Device User.*

***Mode Configuration Extension* Uses Ports Definitions**

None

E.1.3 Modes of Service

Not applicable.

E.1.4 Service States

There are no changes from A. Ethernet Device.

E.1.5 Referenced Documents

There are no changes from A. Ethernet Device

E.2 SERVICES

E.2.1 Provide Services

The *Header Configuration Extension* provides service consists of the following service ports, interfaces, and primitives, which can be called by other client components.

Table 8 – Header Configuration Extension Provide Service Interface

Service Group (Port Name)	**Service (Interface Provided)**	**Primitives (Provided)**
ethernet_header_config_in_port	Ethernet:: EthernetHeaderConfig	getRetainHeader()
		setRetainHeader()

E.2.2 Use Services

None

E.2.3 Interface Modules

E.2.3.1 Ethernet

E.2.3.1.1 EthernetHeaderConfig Interface Description

The *Header Configuration Extension* is shown in the diagram below. The *EthernetHeaderConfig* interface allows a user to configure the Device to retain/accept the ethernet MAC header on packets being pushed to/from the *Device User*.

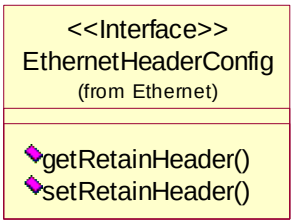

Figure 13 – EthernetHeaderConfig Interface Diagram

E.2.4 Sequence Diagrams

None

E.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section E.5, UML. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

E.3.1 Ethernet::EthernetHeaderConfig

E.3.1.1 *getRetainHeader* Operation

The *getRetainHeader* operation allows the user to determine what configuration the *Device* is currently configured relative to retaining or creating the ethernet MAC header.

E.3.1.1.1 Synopsis

boolean getRetainHeader ();

E.3.1.1.2 Parameters

None

E.3.1.1.3 State

ENABLED CF::Device::operationalState.

E.3.1.1.4 New State

This operation does not cause a state change.

E.3.1.1.5 Return Value

Type	Description
boolean	TRUE = *Device* is set to retain/accept the ethernet MAC header on packets being pushed to/from the *Device User* FALSE = *Device* is set to remove/create the ethernet MAC header on packets being pushed to/from the *Device User*

E.3.1.1.6 Originator

Service User

E.3.1.1.7 Exceptions

None

E.3.1.2 *setRetainHeader* Operation

The *setRetainHeader* operation is used to set the device to either retain or remove the ethernet MAC header.

E.3.1.2.1 Synopsis

void setRetainHeader (in boolean retainHeader);

E.3.1.2.2 Parameters

Parameter Name	Type	Description
retainHeader	Boolean	TRUE = Set the *Device* to retain/accept the ethernet MAC header on packets being pushed to/from the *Device User*; FALSE = Set the *Device* remove/create the ethernet MAC header on packets being pushed to/from the *Device User*

E.3.1.2.3 State

ENABLED CF::Device::operationalState.

E.3.1.2.4 New State

This operation does not cause a state change.

E.3.1.2.5 Return Value

None

E.3.1.2.6 Originator

Service User

E.3.1.2.7 Exceptions

None

E.4 IDL

E.4.1 EthernetHeaderConfigExt

```
/*
** EthernetHeaderConfigExt.idl
*/

#ifndef __ETHERNETHEADERCONFIGEXT_DEFINED
#define __ETHERNETHEADERCONFIGEXT_DEFINED

#ifndef __ETHERNET_DEFINED
#include "Ethernet.idl"
#endif

module Ethernet {

	interface EthernetHeaderConfig {

		boolean getRetainHeader();

		void setRetainHeader(
			in boolean retainHeader
		);
	};
};

#endif // __ETHERNETHEADERCONFIGEXT_DEFINED
```

E.5 UML

This section contains the Device component UML diagram and the definitions of all data types referenced (directly or indirectly) by the Service Primitives and Attributes in E.3.

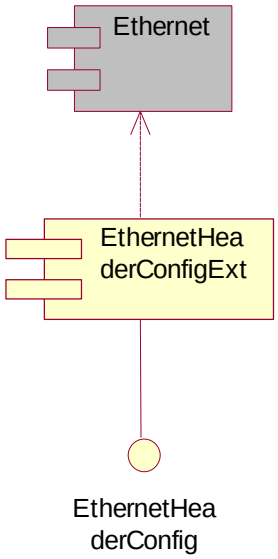

Figure 14 – Header Configuration Extension Component Diagram

E.5.1 Data Types

None

E.5.2 Enumerations

None

E.5.3 Exceptions

None

E.5.4 Structures

None

APPENDIX E.A ABBREVIATIONS AND ACRONYMS

There are no changes from A. Ethernet Device.

APPENDIX E.B PERFORMANCE SPECIFICATION

Table 9 provides a template for the generic performance specification for the *Ethernet Device Header Configuration Extension* API which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 12.

Table 9 – Header Configuration Extension Performance Specification

Specification	Description	Units	Value
Worst Case Command Execution Time for ethernet_hdr_config_in_port	*	*	*

Note: (*) These values should be filled in by individual developers.

F. MAC ADDRESS EXTENSION

F.1 INTRODUCTION

The *Media Access Control* (*MAC) Address Extension* is based upon the *Ethernet Device API*. It extends the functionality of the *Ethernet Device* to allow a user to query the *Ethernet Device* MAC address.

F.1.1 Overview

This extension contains as follows:

a. Section F.1, *Introduction*, contains the introductory material regarding the overview, service layer description, modes, states and referenced documents of this document.

b. Section F.2, *Services*, provides summary of service interface uses, interface for each device component, port connections, and sequence diagrams.

c. Section F.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Ethernet Device* MAC Address Extension.

d. Section F.4, IDL.

e. Section F.5, *UML*.

f. Appendix F.A, *Abbreviations and Acronyms.*

g. Appendix F.B, *Performance Specification.*

F.1.2 Service Layer Description

F.1.2.1 MAC Address Extension Port Connections

Figure 15 shows the port connections for *MAC Address Extension.*

Note: All port names are for reference only. Ports identified in black are provided in A. Ethernet Device.

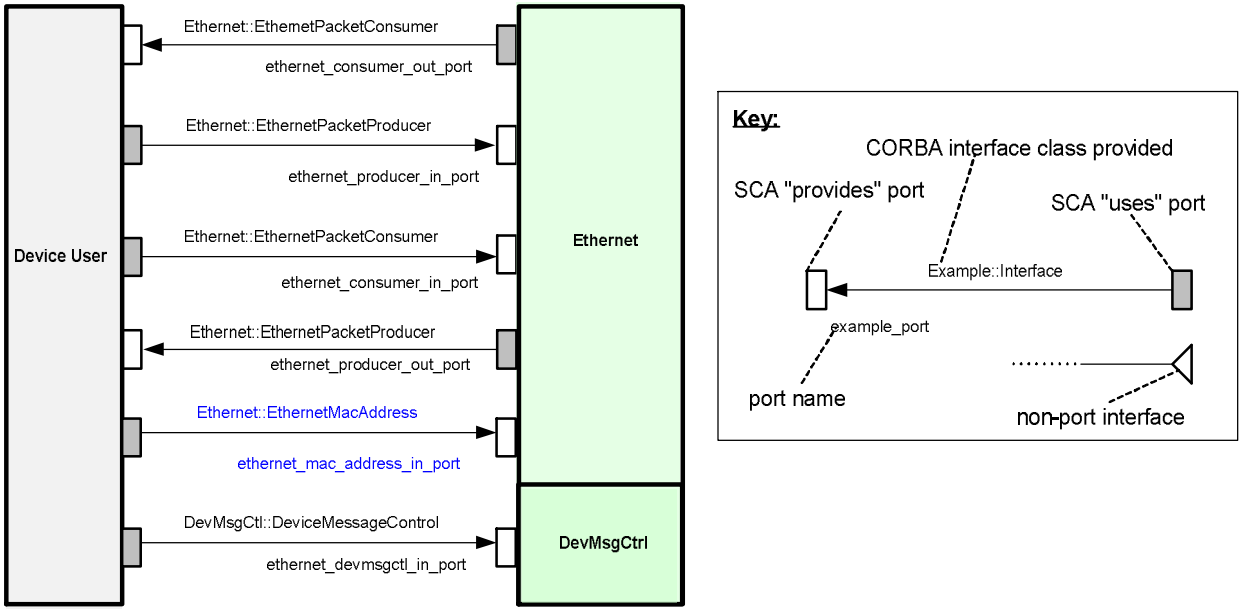

Figure 15 - MAC Address Extension Port Diagram

***Ethernet Mac Address Extension* Provides Ports Definitions**.

ethernet_mac_address_in_port is provided by the *Ethernet Device* to allow a user to query the *Ethernet Device* MAC address.

***Ethernet Mac Address Extension* Uses Ports Definitions**

None

F.1.3 Modes of Service

Not applicable.

F.1.4 Service States

There are no changes from A. Ethernet Device.

F.1.5 Referenced Documents

There are no changes from A. Ethernet Device

F.2 SERVICES

F.2.1 Provide Services

The *MAC Address Extension* provides service consists of the following service ports, interfaces, and primitives, which can be called by other client components.

Table 10 – MAC Address Extension Provide Service Interface

Service Group (Port Name)	**Service (Interface Provided)**	**Primitives (Provided)**
ethernet_mac_address_in_port	Ethernet::EthernetAddress	getMacAddress()

F.2.2 Use Services

None

F.2.3 Interface Modules

F.2.3.1 Ethernet

F.2.3.1.1 EthernetAddress Interface Description

The *MAC Address Extension* is shown in the diagram below. The *EthernetAddress* interface allows a user to query the *Ethernet Device* MAC address.

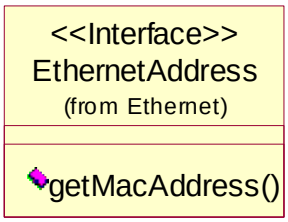

Figure 16 – EthernetAddress Interface Diagram

F.2.3.2 Sequence Diagrams

None

F.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section F.5, UML. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

F.3.1 Ethernet::EthernetAddress

F.3.1.1 *getMacAddress*Operation

The *getMacAddress* operation allows the user to query the *Ethernet Device* MAC address.

F.3.1.1.1 Synopsis

CF::OctetSequence getMacAddress ();

F.3.1.1.2 Parameters

None

F.3.1.1.3 State

ENABLED CF::Device::operationalState.

F.3.1.1.4 New State

This operation does not cause a state change.

F.3.1.1.5 Return Value

Type	Description
CF::OctetSequence (See *SCA* [6])	Returns a sequence of 6 octets which represent the MAC address of the *Ethernet Port Device*.

F.3.1.1.6 Originator

Service User

F.3.1.1.7 Exceptions

None

F.4 IDL

F.4.1 EthernetMacAddressExt

```
/*
** EthernetMacAddressExt.idl
*/

#ifndef __ETHERNETMACADDRESSEXT_DEFINED
#define __ETHERNETMACADDRESSEXT_DEFINED

#ifndef __ETHERNET_DEFINED
#include "Ethernet.idl"
#endif

module Ethernet {

	interface EthernetAddress {

		CF::OctetSequence getMacAddress();

	};
};

#endif // __ETHERNETMACADDRESSEXT_DEFINED
```

F.5 UML

This section contains the Device component UML diagram and the definitions of all data types referenced (directly or indirectly) by the Service Primitives and Attributes in F.3.

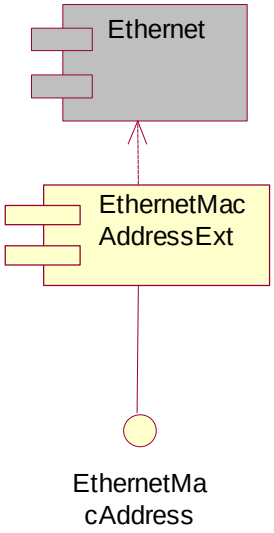

Figure 17 – MAC Address Extension Component Diagram

F.5.1 Data Types

None

F.5.2 Enumerations

None

F.5.3 Exceptions

None

F.5.4 Structures

None

APPENDIX F.A ABBREVIATIONS AND ACRONYMS

There are no changes from A. Ethernet Device.

APPENDIX F.B PERFORMANCE SPECIFICATION

Table 11 provides a template for the generic performance specification for the *Ethernet Device MAC Address Extension* which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 15.

Table 11 – MAC Address Extension Performance Specification

Specification	Description	Units	Value
Worst Case Command Execution Time for ethernet_mac_address_in_port	*	*	*

Note: (*) These values should be filled in by individual developer