Interworking Call Control

Service Definition

1092134 1.6

Interworking Call Control

Service Definition

1092134 1.6

Trillium Digital Systems, Inc.
12100 Wilshire Blvd., Suite 1800
Los Angeles, CA 90025-7118
Phone: +1 (310) 442-9222
Fax: +1 (310) 442-1162
Web: http://www.trillium.com

Interworking Call Control
Service Definition
1092134 1.6

Trillium, Trillium Digital Systems, and TAPA are registered trademarks of Trillium Digital Systems, Inc. Other referenced trademarks are trademarks (registered or otherwise) of the respective trademark owners.

This document is confidential and proprietary to Trillium Digital Systems, Inc. No part of this document may be reproduced, stored, or transmitted in any form by any means without the prior written permission of Trillium Digital Systems, Inc.

Information furnished herein by Trillium Digital Systems, Inc., is believed to be accurate and reliable. However, Trillium assumes no liability for errors that may appear in this document, or for liability otherwise arising from the application or use of any such information or for any infringement of patents or other intellectual property rights owned by third parties which may result from such application or use. The products, their specifications, and the information appearing in this document are subject to change without notice.

To the extent this document contains information related to software products you have not licensed from Trillium, you may only apply or use such information to evaluate the future licensing of those products from Trillium. You should determine whether or not the information contained herein relates to products licensed by you from Trillium prior to any application or use.

Printed in U.S.A.

Copyright 1989-1999 by Trillium Digital Systems, Inc. All rights reserved.

Preface

Objective

This document provides a detailed description of the services provided by the Interworking Call Control software (p/n 1000134) designed by Trillium Digital Systems, Inc. This product is referred to as ICC in the rest of the document.

Audience

Trillium assumes that the readers of this document understand telecommunication protocols, specifically SS7, ATM, and ISDN.

Document Organization

The information in this document is organized into the following sections:

Section	Description
1 Introduction	Provides a textual and graphical overview of the product. It also contains Trillium-specific abbreviations.
2 Environment	Describes the assumptions about the software environment for operating the ICC software
3 Interface Services	Describes in detail the interface primitives at the ICC layer interfaces
4 Interface Procedures	Defines the interface procedures

Document Set

The suggested reading order of this document set is:

1. *Interworking Call Control Functional Specification*

   Contains the features and highlights that describe the protocol and system characteristics of the software. It includes the memory characteristics and conformance details.

2. *Interworking Call Control Training Course*

   Offers a detailed overview of the features and interfaces of the software. It contains code samples, data flow diagrams, and a list of files.

3. *Interworking Call Control Service Definition*

   Describes the procedures and the layer manager interface that are used to pass information between the software and the other software elements. The Interface Primitives section describes the services of the software. The Procedures section describes and illustrates the flow of primitives and messages across the interfaces.

4. *CCT Interface Service Definition*

   This document provides a detailed overview of the services at the CCT interface, which exists between GCC and the protocol-specific interface functions.

5. *Interworking Call Control Interface Service Definition*

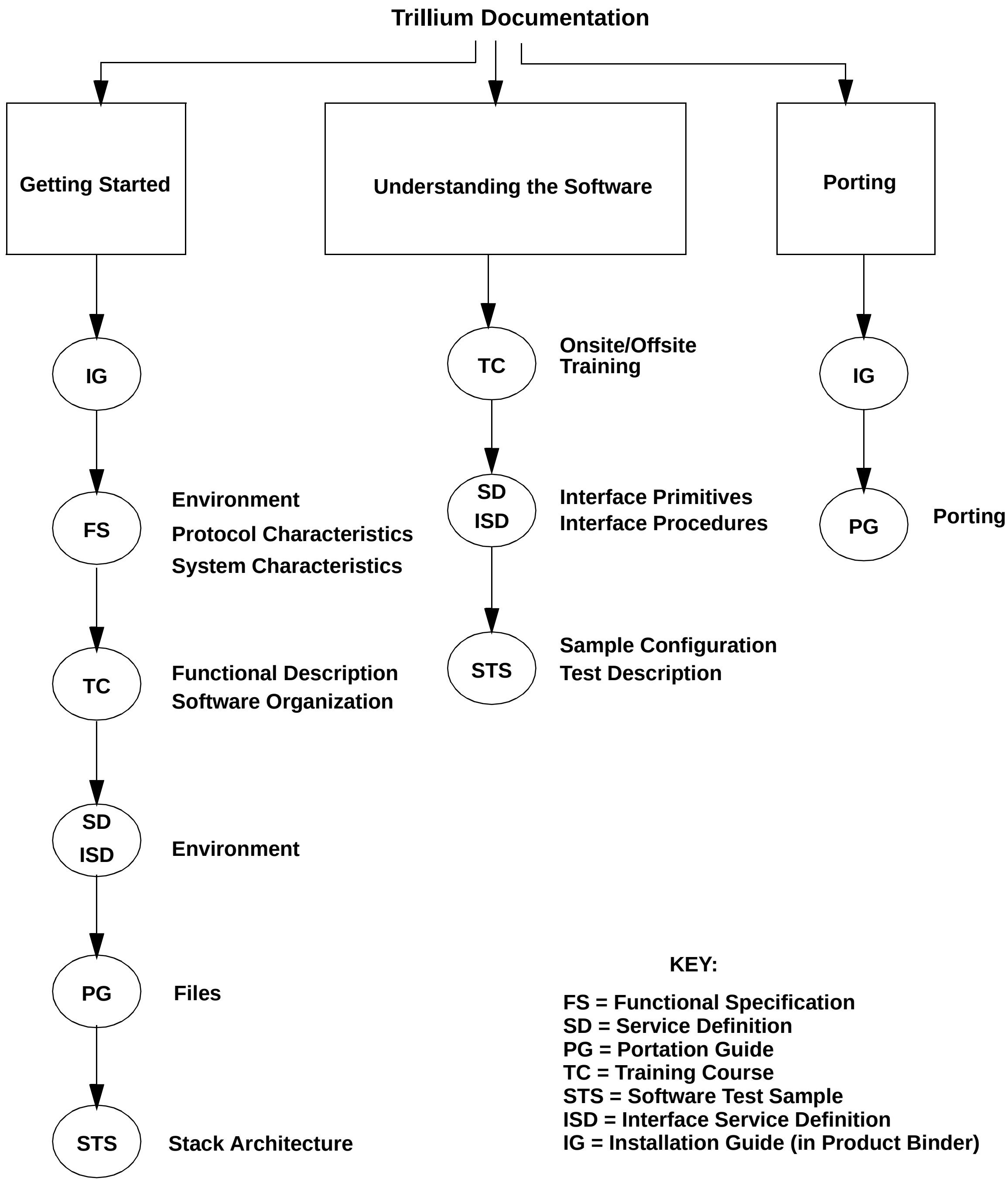
   Provides a detailed overview of the internal interfaces of ICC, namely the RTT, RMT, and SFT interfaces.

6. *Interworking Call Control Portation Guide*

   Describes the files and procedures necessary to port the software to the operating system, into a specific processor family and system architecture. It lists the product, common, and sample files associated with the software.

Using Trillium Documentation

The figure below illustrates the various approaches the user can take when utilizing the software documentation. First time users should read the documents under the **Getting Started** column; important sections and subsections are listed to the right of each document. For users familiar with the documentation but who need to look up certain points concerning the use of the software, the **Understanding the Software** column is suggested. The **Porting** column is for those users who are familiar with Trillium software and related telecommunications protocols and who wish to install the software immediately onto their operating systems.



Notations

This table displays the notations used in this document:

Notation	Explanation	Examples
Arial	**Titles**	**1.1 Title**
Palatino	Body text	This is body text.
Bold	**Highlights information**	**Loose coupling, tight coupling**, **upper layer interface**
ALL CAPS	CONDITIONS, MESSAGES	AND, OR CONNECT ACK
Italics	*Document names, emphasis*	*Interworking Call Control Service Definition* This adds *emphasis*.
`Courier New Bold`	**`Code`** **`Filenames, pathnames`**	**`PUBLIC S16 XxYyIntCfgReq(pst, cfg)`** **`Pst      *pst;`** **`XxMngmt  *cfg;`**

Release History

This table lists the history of changes in successive revisions to this document:

Version	Date	Initials	Description
1.6	12/22/99	rs	• Added information on the ICC Fault-Tolerant/ High-Availability functionality • Conforms to software release 1.5
1.5	09/29/99	mg	• Limited release • Added information supporting the VTOA and Feature Transparency solutions • Conforms to software release 1.4
1.4	05/07/99	mg	• Changes for support of VTOA and feature transparency solutions • Interim release
1.3	02/09/99	rs	• Added information about Q.93B/PNNI support • Conforms to software release 1.2
1.2	10/08/98	rs	• Added information about Q.930/Q.931 support • Conforms to software release 1.1
1.1	07/15/98	ao	Initial release

Contents

Illustrations

1 INTRODUCTION

This document provides the service definition for the Interworking Call Control (ICC) software designed by Trillium Digital Systems, Inc.

This document includes the Layer Manager (LM) descriptions of the Generic Call Control (GCC), Router (RT), Resource Manager (RM), Connection Manager (XM) and Switching Fabric Manager (SFM).

ICC provides the following functionality:

- Starts the protocol stacks
- Maps events from the incoming protocol to the outgoing protocol
- Routes incoming calls
- Interacts with SFM to control the switching fabric
- Allocates resources
- Establishes and de-establishes calls
- Provides overlap and enbloc signalling
- Originates and terminates calls for trunking

Note: *The Protocol-Specific Interface Function (PSIF) is referenced in this document. The PSIF is the highest layer of the underlying protocol and converts the upper interface of the underlying protocol layer to the generic CCT interface.*

2 ENVIRONMENT

This section describes the assumptions about the environment in which ICC is designed to operate.

ICC modules adhere to the Open Systems Interconnection (OSI) reference model that allows standardized procedures to be defined, which also enables the interconnection and subsequent exchange of information between application processes in end systems.

Interaction between ICC, the lower layers, and LM takes place using a set of primitive functions. The primitives either initiate or are the result of the interactions between two layers of the OSI reference model. These primitives—requests, indications, responses, and confirms—completely define the interaction between layers.

ICC has no upper layer, because it is the uppermost layer in the protocol stacks.

ICC consists of the following entities:

- Generic Call Control (GCC)
- Router (RT)
- Resource Manager (RM)
- Connection Manager (XM)
- Switching Fabric Manager (SFM)

Figure 2-1 shows the different interfaces and the interaction between the ICC entities:

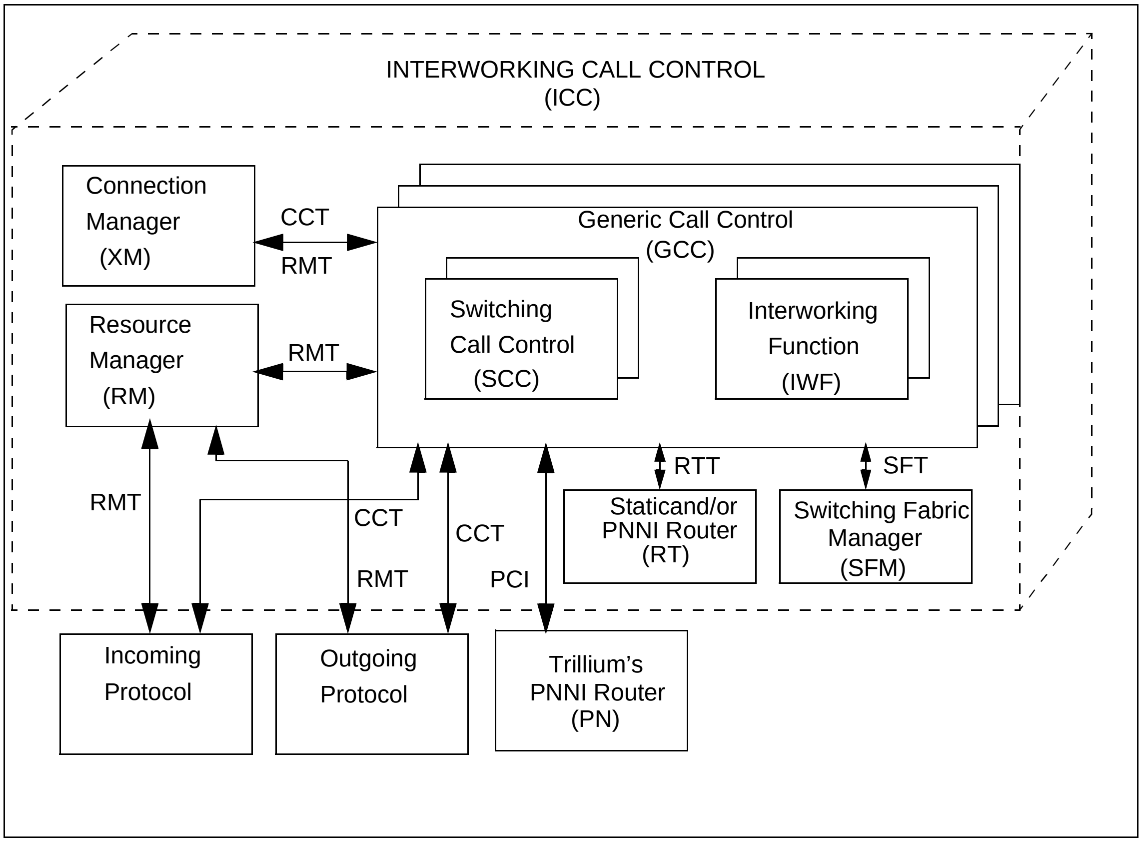


Figure 2-1: ICC stack architecture

ICC has a default RT and RM. The interfaces toward RM, RT, and SFM are clearly defined and can be replaced by a customer-specific implementation.

Some of the entities (such as SFM) may need some porting work to fit the needs of the underlying hardware.

GCC sends requests and responses to the lower layers. The lower layers send indications and confirmations to GCC.

The ICC entities communicate via Service Access Points (SAPs). Figure 2-2 demonstrates the interaction between the ICC entities and their SAPs:

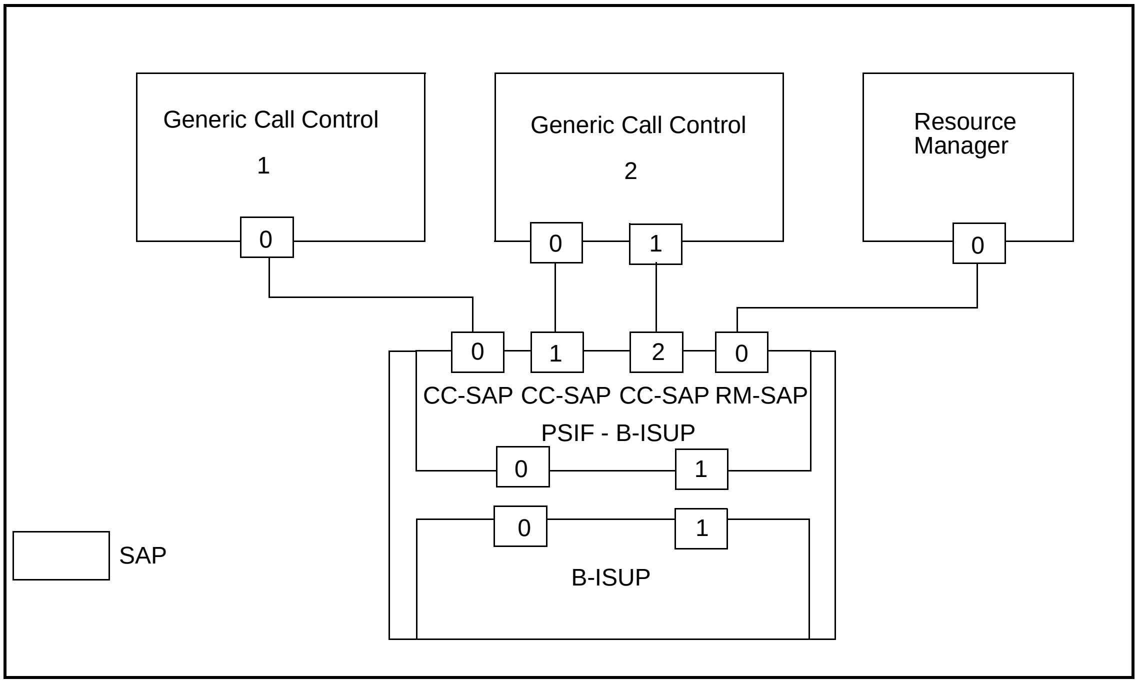


Figure 2-2: Example of ICC SAPs

GCC uses services provided by the underlying entities via Service Access Points (SAPs). A particular service is provided to the upper layer, or received from the lower layer, by the exchange of a sequence of primitives across the SAP. For each variant of the protocol, a different SAP is used.

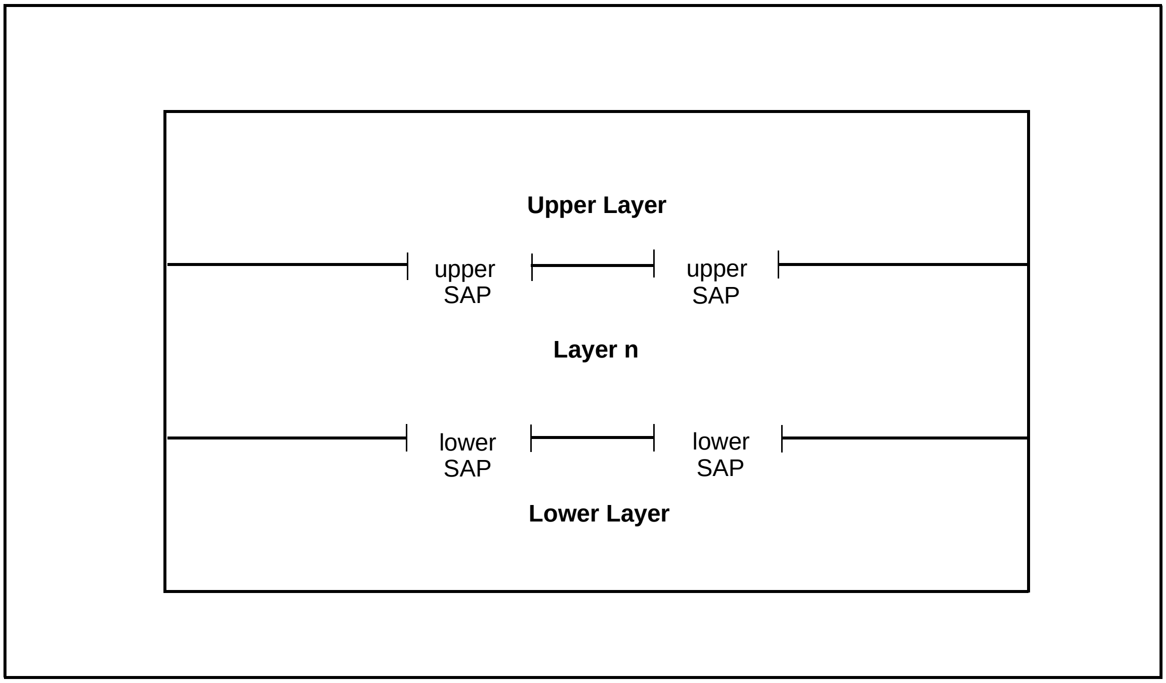


Figure 2-3: Service Access Points (SAPs)

The standardized interface of primitives and SAPs allows layers to be defined independently of each other. As long as the requirements of the layer interface are met, modifications may be made to one entity without affecting any other entity. The standardized interface also allows customers to replace certain entities (such as RT) with their own implementation.

An entity interacts with the upper layer, lower layer, and LM using the primitives and SAPs described above. The entity also interacts with system services by using a simple function interface. Some entities, such as GCC or RT, may not have an upper or lower interface depending on their functionality.

LM provides functions to control and monitor the condition of each protocol layer. It also provides functions to configure default parameters used by ICC.

System services are the functions required by ICC for buffer management, timer management, date and time management, resource checking, and initialization.

In a multiprocessor or multi-tasking system, the different entities are loosely coupled to other entities via queues. In a uniprocessor system, the entities are tightly coupled to other entities via function calls.

3 INTERFACE SERVICES

This section describes in detail the interface primitives at the layer interfaces of ICC.

3.1 General

ICC is the service user for the protocol layers (one or more of B-ISUP, ISUP, ISDN, and Q.93B), RM, RT, SFM, and as an option, XM and PN (Trillium's PNNI router).

As a service user, GCC initially binds itself to the lower layers, that is, it registers itself to the protocol layers, RM, RT, XM, and SFM. In binding, GCC identifies itself and specifies the SAP used.

3.2 Data Types

The sizes of the primitive data types are defined as:

Mnemonic	# of 8-bit bytes	Sign
S8	1	Signed
U8	1	Unsigned
S16	2	Signed
U16	2	Unsigned
S32	4	Signed
U32	4	Unsigned
PTR	as required	Unsigned

The size of **PTR** depends on the specific machine to which the software is ported.

The following table contains information on the **typedef**s used.

New Data Type	Data Type	Purpose
Bool	**U8**	Boolean
Cntr	**S32**	Statistics counter
VcId	**U16**	Virtual connection ID
VpId	**U16**	Virtual path ID
SwtchIdx	**U32**	Switching index of the connection as maintained by the SFM
Vcci	**U16**	Virtual channel connection ID
CID	**U8**	Channel ID in a VCCI
Operation	**U8**	Switching fabric operation
Direction	**U8**	Call direction

3.3 Common Structures

Each management primitive consists of a header and a status field, which is followed by a structure specific to the type of primitive invoked. The primitive-specific part of the management structure is described with each primitive.

3.3.1 Header

Header structure has the following format:

```
typedef struct tds_header
{
   U16 msgLen;                  /* message length      - not used */
   U8 msgType;                  /* message type        - used always */
   U8 version;                  /* version            - not used */
   U16 seqNmb;                  /* sequence number     - not used */
   EntityId entId;              /* entity id         - used always */
   ElmntId elmId;            /* element id         - used sometimes */
   TranId transId;              /* transaction Id      - mandatory */
   Resp response;               /* response parameters - mandatory */
} Header;
```

msgLen

Message length. It is not used.

msgType

Message type. The allowable values for GCC are:

Message type	Description
TCFG	Configuration
TCNTRL	Control
TSSTA	Solicited status
TSTS	Statistics
TUSTA	Unsolicited status
TTRC	Trace indication
TACNT	Accounting indication

The allowable values for RT are:

Message type	Description
TCFG	Configuration
TCNTRL	Control
TSTS	Statistics
TSSTA	Solicited status
TUSTA	Unsolicited status

The allowable values for the RM are:

Message type	Description
TCFG	Configuration
TCNTRL	Control
TSTS	Statistics
TSSTA	Solicited status
TUSTA	Unsolicited status

The allowable values for the SFM are:

Message type	Description
TCFG	Configuration
TCNTRL	Control
TUSTA	Unsolicited status

The allowable values for the XM are:

Message type	Description
TCFG	Configuration
TCNTRL	Control
TSTS	Statistics
TSSTA	Solicited status
TUSTA	Unsolicited status

version

Version. It is not used.

seqNmb

Sequence number. This field is significant only when a corresponding confirm class of primitives responds to the layer manager request; that is, when a status request is answered with a status confirm. In such cases, the **seqNmbr** in the confirm primitives is the same as that received in the corresponding request primitive.

entId

Structure entity ID. It has the following format:

```
typedef struct entityId
{
   Ent ent;
   Inst inst;
} EntityId;
```

ent

Entity. The allowable value for GCC is **ENTCC**.

inst

Entity instance.

elmId

Structure element ID. It has the following format:

```
typedef struct elmntId            /* element id */
{
   Elmnt elmnt;
   ElmntInst1 elmntInst1;
   ElmntInst2 elmntInst2;
   ElmntInst3 elmntInst3;
} ElmntId;
```

elmnt

Element. The allowable values for GCC are:

Element	Description
STGEN	General
STCCPSSAP	Protocol SAP
STCCRMSAP	Resource Manager (RM) SAP
STCCRTSAP	Router SAP, for the static router and Trillium’s PNNI router
STCCSFSAP	Switching Fabric Manager (SFM) SAP

Element	Description
STCCINTFC	Interface configuration, control, status, and / or statistics
STCCPROF	ATM profile configuration and / or control
STCCOBS	Observation trigger table configuration, control, and / or status
STCCVINTFC	Virtual interface configuration, control, status, and / or statistics
STSID	System ID
STCCTSTCALL	Continuity check test call
STCCCDR	Call detail record

The allowable values for the RT are:

Element	Description
LRT_GEN	General
LRT_SAP	Router CC SAP
LRT_ROUTE	Route
LRT_INTF	Interface
LRT_OBS	Observation trigger index
LRT_CONG	Congestion control
LRT_VINTF	Virtual interface
STRTAUDPAP	Periodic Audit Procedure (PAP) auditing request
STRTAUDOAP	One-time Audit Procedure (OAP) auditing request
STGRRTSAP	Group SAP

The allowable values for the RM are:

Element	Description
STGEN	General
STRMUPSAP	Upper SAP
STRMBBPHY	Physical broadband link
STRMVP	Broadband resource VPI
STRMVC	Broadband resource VCI
STRMBBINTFC	Broadband interface
STRMNBDPC	Narrowband interface DPC
STRMCIC	Narrowband resource circuit
STRMDSS1INTFC	DSS1 interface
STRMPVC	Static binding between resources
STRMAUDPAP	PAP auditing request
STRMAUDOAP	OAP auditing request
STRMAUDGAP	GCC audit procedure (GAP) auditing request
STGRRMSAP	Group SAP
STRMUPSAP	SAP

The allowable values for the SFM are:

Element	Description
LSF_GEN	General description
LSF_SAP	Upper SAP configuration

The allowable values for the XM are:

Element	Description
STGEN	General
STXMCCSAP	Connection Management Upper SAP
STXMRMSAP	Resource Management Upper SAP
STXMFEATTRPIWF	Feature transparency IWF control block
STXMPH1TKIWF	Phase 1 trunking IWF control block
STXMPH2TKIWF	Phase 2 trunking IWF control block
STXMPH2RSCCB	Phase 2 ATM resource control block

Element	Description
STXMSIGVCCI	SIGVCCI control block
STXMVCCI	VCCI control block
STXMCID	CID control block
STXMCIC	CIC control block
STXMATMPROF	ATM profile configuration
STXMVTOAPROF	VTOA profile configuration

Some of the values are relevant only for certain options. For example, the broadband fields are relevant only when using B-ISUP.

elmntInst1, elmntInst2, elmntInst3

Element instance. It is not used.

transId

Transaction ID. The layer manager uses this value to correlate confirm messages with request messages when more than one outstanding request awaits confirmation. When GCC receives a request and generates a confirmation, it must copy the value of this field from the request message to the confirm message.

response

The response information is used when GCC, RT, SFM, or RM sends the confirmation for a layer manager primitive. The response is sent to the source that sent the request. The source entity and instance are taken from the post structure. The response field contains additional information that the layer requires in order to send the confirmation to the caller.

```
typedef struct resp
{
   Selector selector;           /* selector */
   Priority prior;              /* priority */
   Route route;                 /* route */
   MemoryId mem;                /* memory */
}Resp;
```

selector

The layer uses this selector value when sending the confirmation. This value is generated by the sender of the request.

prior

The layer uses this priority value when sending the confirmation. This value is generated by the sender of the request.

route

The layer uses this route value when sending the confirmation. This value is generated by the sender of the request.

mem

See Section 3.3.5, "**Memory**."

3.3.2 Status

The status field indicates the result of a request. This information is valid in the confirm primitives only. The status field has the following format:

```
typedef struct cmStatus
{
   U16   status;           /* Status of the operation  */
   U16   reason;           /* If failed, the reason    */
} CmStatus;
```

status

This field is used to return the status of the requested primitive (for example, **LCM_PRIM_OK** or **LCM_PRIM_NOK**) to indicate whether the primitive succeeded or failed. In some cases, the primitive's processing is deferred, so the result (success or failure) of the processing is not immediately available. In such cases, **LCM_PRIM_OK_NDONE** is returned indicating that the primitive was received, but that processing has been deferred. After processing is complete, **LCM_PRIM_OK** or **LCM_PRIM_NOK** is returned in the normal case. This deferred processing (and **LCM_PRIM_OK_NDONE**) applies to control requests only.

reason

This field contains the cause of the failure. The range of values from 0 to 255 is used for general failure codes. Values from 256 onward indicate protocol-specific reasons for failure.

If the request was successful, the status **LCM_PRIM_OK** is returned. In this case, the **reason** has no significance and is set to **LCM_REASON_NOT_APPL**. The status is of interest only when the request failed. For example, if the entity ID received in a configuration request does not match the entity ID of the layer (receiver), the receiver sends a confirmation with the following status information:

```
status = LCM_PRIM_NOK
reason = LCM_REASON_INVALID_ENTITY
```

3.3.3 Pst

All primitives have the post structure as their first parameter. The post structure routes the primitive from the source layer to the destination layer.

pst->selector determines the correct interface function called when resolving the primitive at the calling layer. It determines the memory region and pool from which message buffers are allocated, as well as the priority and route for the message. It also specifies the source and destination entities.

Once the primitive reaches the destination layer, this parameter is no longer useful except in the bind request.

When a source layer generates a primitive at an SAP, that primitive must be routed to the destination layer. The following **pst** structure provides the information to route the primitive.

```
typedef struct pst              /* parameters for SPstTsk */
{
   ProcId dstProcId;            /* destination processor id */
   ProcId srcProcId;            /* source processor id */
   Ent dstEnt;                  /* destination entity */
   Inst dstInst;                /* destination instance */
   Ent srcEnt;                  /* source entity */
   Inst srcInst;                /* source instance */
   Prior prior;                 /* priority */
   Route route;                 /* route */
   Event event;                 /* event */
   Region region;               /* region */
   Pool pool;                   /* pool */
   Selector selector;           /* selector */
   U16 spare1;                  /* spare for alignment */
} Pst;
```

The **dstProcId**, **dstEnt**, and **dstInst** identify the destination (called) layer.

The **srcProcId**, **srcEnt**, and **srcInst** identify the source (calling) layer.

The **priority** and **route** identify the message priority and route used for the messages sent at this SAP.

event identifies the primitive type. The packing function of the calling layer initializes this value only in the case of loose coupling. The unpacking function of the called layers uses this value to decode the received message into the appropriate primitive.

The **region** and **pool** identify the dynamic memory pool from which the source layer allocates messages, when required, for communicating with the destination layer.

selector identifies the specific interface coupling function invoked to resolve this primitive.

spare1 aligns the structure on a 32-bit bandwidth.

3.3.4 Timer Configuration

The timer configuration has the structure:

```
typedef struct tmrCfg
{
   Bool enb;        /* enable */
   U16  val;        /* value */
} TmrCfg;
```

enb

Boolean. This indicates whether the timer is enabled.

val

Provides the value of the timer in ticks.

3.3.5 Memory

Identifies the memory region and pool ID from which the buffers are allocated for packing primitives sent across loosely coupled interfaces.

```
typedef struct memoryId
{
   Region region;        /* region */
   Pool   pool;          /* pool */
} MemoryId;
```

3.3.6 Interface

This data structure defines an interface.

```
typedef struct rmInterface
{
   U8  intfType;
   union interface
   {
      Dpc         dpc;       /* For ISUP, B-ISUP Interface type */
      U32         intfId;    /* DSS1, DSS2  Interfaces */
   } t;
} RmInterface;
```

`intfType`

Identifies the type of interface. The following values are possible:

Value	Description
`CC_BI_INTFC`	B-ISUP interface type
`CC_SI_INTFC`	ISUP interface type
`CC_IN_INTFC`	ISDN (DSS1 interface type)
`CC_AM_INTFC`	Q.93B / PNNI (DSS2 interface type)
`CC_FEATTRP_INTFC`	Feature transparency interface type
`CC_PH1TK_INTFC`	VTOA phase 1 AAL1 or AAL2 trunking interface type
`CC_PH2TK_INTFC`	VTOA phase 2 AAL1 or AAL2 trunking interface type
`CC_PHY_TK_INTFC`	Physical trunking interface

`dpc`

Identifies the destination signalling point code. This field is valid in the case of ISUP and B-ISUP signalling.

`intfId`

Identifies a unique interface in the domain identified by the **`intfType`**. This field is valid for DSS1, DSS2, and all trunking interfaces.

3.3.7 Protocol Variants

ICC supports the following lists of protocol variants.

B-ISUP Protocol Variants

Variant	Description
CC_BIITU	B-ISUP ITU protocol variant
CC_BIATF	B-ISUP ATM Forum protocol variant

ISUP Protocol Variants

Variant	Description
CC_SIITU92	ISUP ITU 92 protocol variant
CC_SI76792	ISUP Q.767 92 protocol variant
CC_SIANS92	ISUP ANSI 92 protocol variant
CC_SIETSI	ISUP ETSI protocol variant

ISDN Protocol Variants

Variant	Description
CC_INQSIG	ISDN QSIG
CC_INETSI	ISDN ETSI
CC_INNI2	ISDN NI2
CC_INITU	ISDN ITU

Q.93B Protocol Variants

Variant	Description
CC_AM_SIG_PNNI	Q.93B PNNI protocol variant
CC_AM_Q2931	Q.93B Q.2931 protocol variant
CC_AM_UNI40	Q.93B UNI40 protocol variant
CC_AM_UNI31	Q.93B UNI3.1 protocol variant

Feature Transparency Variants

Variant	Description
CC_FEATTRP_SI	ISUP interface for feature transparency
CC_FEATTRP_IN	ISDN interface for feature transparency

VTOA Phase 1 Variants

Variant	Description
CC_PH1TK_AAL1	VTOA phase 1 trunking using AAL1
CC_PH1TK_AAL2	VTOA phase 1 trunking using AAL2

VTOA Phase 2 Variants

Variant	Description
CC_PH2TK_AAL1	VTOA phase 2 trunking using AAL1
CC_PH2TK_AAL2	VTOA phase 2 trunking using AAL2

3.3.8 Network Resource

This event identifies a network resource. The resource type depends on the protocol used.

```
typedef struct rmRsc         /* Generic Resource Structure */
{
   RmInterface intfc;        /* Interface on which resource identified */
   Bool  rscPres;            /* True if the Resource has been Identified */
   union rsc
   {
       RmBbRsc    bbRsc;     /* Broadband Resource */
       RmNbRsc    nbRsc;     /* Narrowband Resource */
       RmDss1Rsc  dss1Rsc;   /* DSS1 Resource */
       RmBbPh1TrnkRsc  bbPh1TrnkRsc; /* ATM phase1 trunking Resource */
       RmBbPh2TrnkRsc  bbPh2TrnkRsc; /* ATM phase2 trunking Resource */
       RmfeatTrpRsc    featTrpRsc;   /* Feature transparency Resource */
       RmAtmTrnkRsc    atmTrnkRsc;   /* ATM AAL1/AAL2 trunk Resource */
   }t;
}RmRsc;
```

intfc

Identifies the interface at which the resource is defined. For more details, see Section 3.3.6, "Interface."

rscPres

This field identifies whether the resource information, **bbRsc** or **nbRsc**, is valid. For the configuration request, this must always be set to TRUE.

bbRsc

Broadband resource. This field must be filled when the **intfc** identifies a **CC_BI_INTFC** or **CC_AM_INTFC** interface type. This structure contains the VPI and VCI of the broadband resource.

```
typedef struct rmBbRsc       /* Broadband Resource */
{
   U8    flag;               /* Flag */
   VpId  vpId;               /* VPI */
   VcId  vcId;               /* VCI */
} RmBbRsc;
```

flag

This field indicates whether the **VpId** and / or **VcId** are / is valid in this resource.

Type	Description
RMT_VPIVCI_SPEC	Both the VPCI / VCI are specified
RMT_VCI_REQDVPI	Specified, and the VCI is required
RMT_VPI_REQD	The VPI should be allocated
RMT_VPIVCI_REQ	Similar to the **rscPres** set to FALSE

nbRsc

Narrowband resource. This field must be filled when the **intfc** identifies an ISUP interface type. This structure contains the Circuit Identification Code (CIC) value of the narrowband resource.

```
typedef struct  rmNbRsc      /* Narrowband Resource */
{
   Cic   cic;                /* Circuit Identification Code */
} RmNbRsc;
```

dss1Rsc

DSS1 resource. This field must be filled when the **intfc** identifies an ISDN interface type. This structure contains the DSS1 channel associated with the DSS1 interface. Refer to the *INT Interface Service Definition* for more details.

```
typedef struct rmDss1Rsc        /* channel id tokens */
{
   ElmtHdr eh;                  /* element header */
   TknU8   infoChanSel;         /* information channel selection */
   TknU8   dChanInd;            /* d channel indicator */
   TknU8   prefExc;             /* preferred/exclusive */
   TknU8   intType;             /* interface type */
   TknU8   intIdentPres;        /* interface identifier present */
   TknU16  intIdent;            /* interface identifier */
   TknU8   chanMapType;         /* channel type/map type */
   TknU8   nmbMap;              /* number/map */
   TknU8   codeStand1;          /* coding standard */
   TknStrM chanNmbSlotMap;      /* channel number/slot map */
} RmDss1Rsc;
```

bbPh1TrnkRsc

Phase 1 AAL1 / AAL2 trunking resource. This field must be filled when the **intfc** identifies a Phase 1 trunking interface type. This structure contains the ATM VCCI value for the Phase 1 AAL1 / AAL2 trunking resource.

```
typedef struct rmBbPh1TrnkRsc    /* ATM Phase1 Trunking Resource */
{
   U8    vcciType;        /* type of VCCI */
   Vcci  vcci;            /* VCCI value */
} RmBbPh1TrnkRsc;
```

vcciType

Flag indicating the VCCI type at the AAL1 / AAL2 Phase 1 trunking interface.

VCCI Type	Description
LXM_SIGVCCI_OVERAAL5	Signalling VCCI over AAL5
LXM_SIGVCCI_OVERAAL2	Signalling VCCI over AAL2
LXM_BEARERVCCI_AAL2	AAL2 bearer VCCI
LXM_BEARERVCCI_AAL1_SINGLE	One-to-one AAL1 bearer VCCI
LXM_BEARERVCCI_AAL1_MUX	Many-to-one AAL1 bearer VCCI

vcci

VCCI of this resource, for the AAL1 / AAL2 Phase1 trunking interface.

bbPh2TrnkRsc

Phase 2 AAL1 / AAL2 trunking resource. This field must be filled when the **intfc** identifies a Phase 2 trunking interface type. This structure contains the signalling correlation tag for the Phase 2 AAL1 / AAL2 resource.

```
typedef struct rmBbPh2TrnkRsc    /* ATM Phase2 Trunking Resource */
{
  SCT   sct;                     /* signalling correlation tag */
  U8    vcciType;                /* type of VCCI */
  Vcci  vcci;                    /* VCCI value */
} RmBbPh2TrnkRsc;
```

sct

Signalling correlation tag of this resource for the AAL1 / AAL2 Phase 2 trunking interface.

vcciType

Flag indicating the VCCI type at the AAL1 / AAL2 Phase 2 trunking interface.

VCCI Type	Description
LXM_SIGVCCI_OVERAAL5	Signalling VCCI over AAL5
LXM_BEARERVCCI_AAL2	AAL2 bearer VCCI
LXM_BEARERVCCI_AAL1_SINGLE	One-to-one AAL1 bearer VCCI
LXM_BEARERVCCI_AAL1_MUX	Many-to-one AAL1 bearer VCCI

vcci

VCCI of this resource for the AAL1 or AAL2 Phase 2 trunking interface.

featTrpRsc

Feature transparency resource. This field must be filled when the **intfc** identifies a feature transparency interface type. This structure contains the virtual channel of the feature transparency resource.

```
typedef struct  rmFeatTrpRsc   /* Narrowband Trunking Resource */
{
   Vcci  vcci;                 /* VCCI value */
   CID   cid;                  /* CID value */
} RmFeatTrpRsc;
```

vcci

The **vcci** of this resource for the feature transparency interface.

cid

The **cid** of this resource for the feature transparency interface.

atmTrnkRsc

Physical ATM trunking resource. This resource is filled when the actual ATM resource associated with a Phase 1 or Phase 2 AAL1 / AAL2 trunking interface must be specified. This structure contains the VPI and VCI associated with the AAL1 / AAL2 VCCI and the AAL1 / AAL2 Channel ID (CID), if applicable.

```
typedef struct rmAtmTrnkRsc    /* Broadband trunking Resource */
{
   U8    flag                  /* flag */
   VpId  vpId;                 /* VPI */
   VcId  vcId;                 /* VCI */
   CID   cid;                  /* AAL1/AAL2 channel id */
} RmAtmTrnKRsc;
```

flag

This field indicates whether the CID field is valid.

CID Field	Description
RMT_CID_INVALID	The CID specified is invalid.
RMT_CID_VALID	The CID specified is valid.

vpId

The ATM **vpId** associated with this Phase 1 or Phase 2 trunking interface.

vcId

The ATM **vcId** associated with this Phase 1 or Phase 2 trunking interface.

cid

The AAL1/AAL2 Channel ID, if applicable. This field is valid if the VPI/VCI corresponds to an AAL2 connection, or a many-to-one AAL1 VCCI.

3.3.9 Route Structure

```
typedef struct rtRoute
{
   U8          addrInd;         /* type of address */
   U8          nmbDigits;       /* number of digits */
   U8          numPlan;         /* numbering plan */
   U8          ident;           /* identification */
   Addrs       addr;            /* routing part of the address */
} RtRoute;
```

addrInd

Identifies the address type. The values are:

Value	Description
CC_CDPTY	Called party number
CC_TRANNET	Transit network selection

nmbDigits

Number of digits in the route. This field identifies the number of digits in the **addr** field. It contains the number of digits and not the number of valid octets compared against in the **addr** field, because the digits are stored in the **addr**, in BCD format.

numPlan

If the **addrInd** indicates the called party number, this field contains the numbering plan identification. In the case of transit network selection, this field contains the network identification plan.

The values of the numbering plan identification are:

Values	Description
CC_NP_UNK	Number not present
CC_ISDNNUM	ISDN numbering plan (CCITT E.164)
CC_DATANUM	Data numbering - X.121
CC_TELEXNUM	Telex numbering - Recommendation F.69
CC_PRIVATENUMPLAN	Private numbering pan
CC_UNKNOWNPLAN	Unknown plan
CC_TELEPNUMPLAN	Telephony numbering plan (CCITT E.163)
CC_NSAPNUMPLAN	NSAP Address for ATM

The values of the network identification plan are:

Values	Description
CC_NI_UNKNWN	Unknown NI plan
CC_NI_DNIC_X21	Public Data Network Identification Code (DNIC), recommendation X.121.
CC_NI_MNIC_E212	Public land Mobile Network ID Code (MNIC), recommendation E.211

ident

If the **addrInd** field indicates that the route is of type called party number, the **ident** corresponds to the nature of the address indicator. If the **addrInd** field indicates that the route corresponds to the transit network selection, this field contains the network identification field type.

The values of the nature of address indicator are:

Values	Description
CC_NA_SUBSNUM	Subscribe number
CC_NA_UNKNOWN	Unknown
CC_NA_NATNUM	National number
CC_NA_INTNATNUM	International number
CC_NA_NSPNUM	NSAP number

addr

The string of digits corresponding to the route or to the transit network selection digits. The digits are stored as BCD digits, with two BCD digits packed in one octet.

3.3.10 Traffic Descriptor

The traffic descriptor contains the information required by the RM, RT, SFM, and Connection Manager (XM).

```
typedef struct  rmTfcDesc
{
  CcNBTfcDesc nbTfcDesc;  /* Narrowband traffic descriptor */
  AalConParam  cacInfo;   /* Broadband Parameter Required to identify the
                             Traffic Requirements */
 } RmTfcDesc;
```

nbTfcDesc

This contains the real-time information associated with an ISUP bearer channel of an interworking call. If the incoming interface of an interworking call is ISUP, GCC derives this information from the incoming ISUP connect event. If the outgoing interface for an interworking call is ISUP, GCC initializes this information from a configuration profile. GCC passes the real-time information to the SFM, which requires this information to determine the switching characteristics of the narrowband bearer channel.

```
typedef struct ccNbTfcDesc       /* Narrowband Traffic Descriptor */
{
   TknU8   trnMedReq;            /* Transmission Medium Requirement */
   TknU8   satInd;               /* Satellite Indicator */
   TknU8   contChkInd;           /* continuity check indicator */
   TknU8   echoCntrlDevInd;      /* echo control device indicator */
   TknU8   cgPtyCat;             /* calling party category */
   TknU32  propDelay;            /* Propagation delay */
} ccNbTfcDesc;
```

trnMedReq

Specifies the requirements of a narrowband bearer channel. The allowable values are:

TMR_SPEECH
TMR_64KBITS
TMR_31KHZ

satInd

Specifies the satellite indicator associated with a narrowband bearer channel. The allowable values are:

SAT_NONE
SAT_ONE
SAT_TWO
SAT_THREE

contChkInd

Specifies the continuity check indicator associated with a narrowband bearer channel. The allowable values are:

RM_COT_NOK
RM_IN_COT_OK
RM_OUT_COT_OK

echoCntrlDevInd

Specifies the echo control device indicator associated with a narrowband bearer channel. The allowable values are:

RM_ECHOCNTRL_NOK
RM_IN_ECHOCNTRL_OK
RM_OUT_ECHOCNTRL_OK

cgPtyCat

Specifies the calling party category associated with a narrowband bearer channel. The allowable values are:

CAT_UNKNOWN
CAT_OPLANGFR
CAT_OPLANGENG
CAT_OPLANGGER
CAT_OPLANGRUS
CAT_OPLANGSP
CAT_ADMIN1
CAT_ADMIN2
CAT_ADMIN3
CAT_ORD
CAT_PRIOR
CAT_DATA
CAT_TEST
CAT_PAYPHONE

propDelay

Specifies the propagation delay associated with a narrowband circuit.

cacInfo

The RM requires the **cacInfo** to perform the CAC algorithm, in the case of an ATM protocol. The XM uses the traffic descriptor to pass the ATM connection parameters associated with an ISUP resource, in the case of Phase 2 AAL1 / AAL2 trunking to GCC. The traffic descriptor is also passed to the SFM, which requires this information to apply the policy function on the bearer channel. The **typedef** of the **AalConParam** is defined below (refer to **cm_atm.x**). For more information, refer to the *BIT Interface Service Definition*.

```
typedef struct        aalConParam   /* connection parameters for AAL */
{
   AmAalParam          aalParam;    /* AAL Parameters */
   AmAtmTfcDesc        atmTfcDesc;  /* ATM Traffic Descriptor */
   AmBBearCap          bBearCap;    /* Broadband Bearer Capability */
   AmQosParam          qosParam;    /* Qos parameters */
   AmEtoeDly           etoeDly;     /* End to End Transit Delay */
   AmOamTfcDesc        oamTfcDesc;  /* OAM Traffic Descriptor */
#if (DEF_SIG_PNNI | DEF_UNI40)
   AmAltAtmTfcDesc     altAtmTfcDesc;  /* Alternative ATM Traffic
                                          descriptor */
   AmMinAccAtmTfcDesc  minAccAtmTfcDesc;
                                    /* Minimum acceptable ATM Traffic
                                          Descriptor */
   AmExtQosParam       extQosParam;    /* Extended QOS parameter */
   AmAbrSetupParam     abrSetupParam;  /* ABR setup parameters */
   AmAbrAddParam       abrAddParam;    /* ABR additional parameters */
#endif /* DEF_SIG_PNNI | DEF_UNI40 */
} AalConParam;
```

3.3.11 Cause

```
typedef struct ccCause
{
   ElmtHdr eh;                 /* element header */
   TknU8   cdeStand;           /* coding standard */
   TknU8   recommend;          /* recommendation */
   TknU8   location;           /* location */
   TknU8   causeVal;           /* cause value */
   TknStrM dgnVal;             /* Diagnostics value */
} CcCause;
```

eh

Element header.

cdeStand

Coding standard.

recommend

Recommendation.

location

Location.

causeVal

Cause value.

dgnVal

Diagnostics value.

Description:

The cause is a protocol-independent cause structure.

3.4 Concepts

3.4.1 Notes on Observation Triggers

GCC maintains an observation trigger table. Each row in the observation trigger table represents incoming parameters in which the layer manager sets an observation. The incoming parameters are the calling party number and incoming resource. If a set of calling party addresses and incoming resources require similar observation, then only one row can represent the whole set.

Each column in the observation trigger table represents incoming parameters in which the layer manager sets an observation. The outgoing parameters are the called party number and outgoing resource. If a set of called party addresses and outgoing resources require similar observation, then only one column can represent the whole set. This concept of observation triggers requires the PSIF, RT, and RM to collaborate with GCC.

The following list contains information on observation trigger tables.

- The observation trigger table resides in GCC.
- Each calling party number observed in the PSIF has an observation index configured with it. This observation index represents a row in the observation trigger table.
- Each called party number observed in the router has an observation index configured with it. This observation index represents a column in the observation trigger table.
- Each incoming interface observed in the RM has an observation index configured with it. This observation index represents a row in the observation trigger table.
- Each incoming resource observed in the RM has an observation index configured with it. This observation index represents a row in the observation trigger table.
- Each outgoing interface observed in the RM has an observation index configured with it. This observation index represents a column in the observation trigger table.
- Each outgoing resource observed in the RM has an observation index configured with it. This observation index represents a column in the observation trigger table.
- Row 0 in the observation trigger table handles defaults.
- Column 0 in the observation trigger table handles defaults.

If an observation is set on a particular calling party number, the layer manager must perform one of the operations in the following scenarios.

1. If there already exists a row with the same observations, then the layer manager configures the same observation row index with the calling party number in the PSIF of the interface, through which a call having this particular calling party may enter.

   Or:

2. If a row does not exist with the similar observation, then the layer manager must configure a new row in the observation trigger table. The layer manager configures the same observation row index with the calling party number in the PSIF of the interface, through which a call having this particular calling party number can enter.

Note: *Column[0] in this row should be set to the default observation type on this particular calling party number.*

If an observation is set on a particular called party number, the layer manager must perform one of the operations in the following scenarios.

1. If there already exists a column with the same observations, the layer manager configures the same observation column index having the called party number in the router.

   Or:

2. If a column does not exist with the similar observation, then the layer manager must configure a new column in the observation trigger table. The layer manager configures the same observation column index with the called party number in the router.

Note: *Row[0] in this column should be set to the default observation type on this particular called party number.*

Figure 3-1 illustrates a scenario of the columns and rows of the observation trigger table.

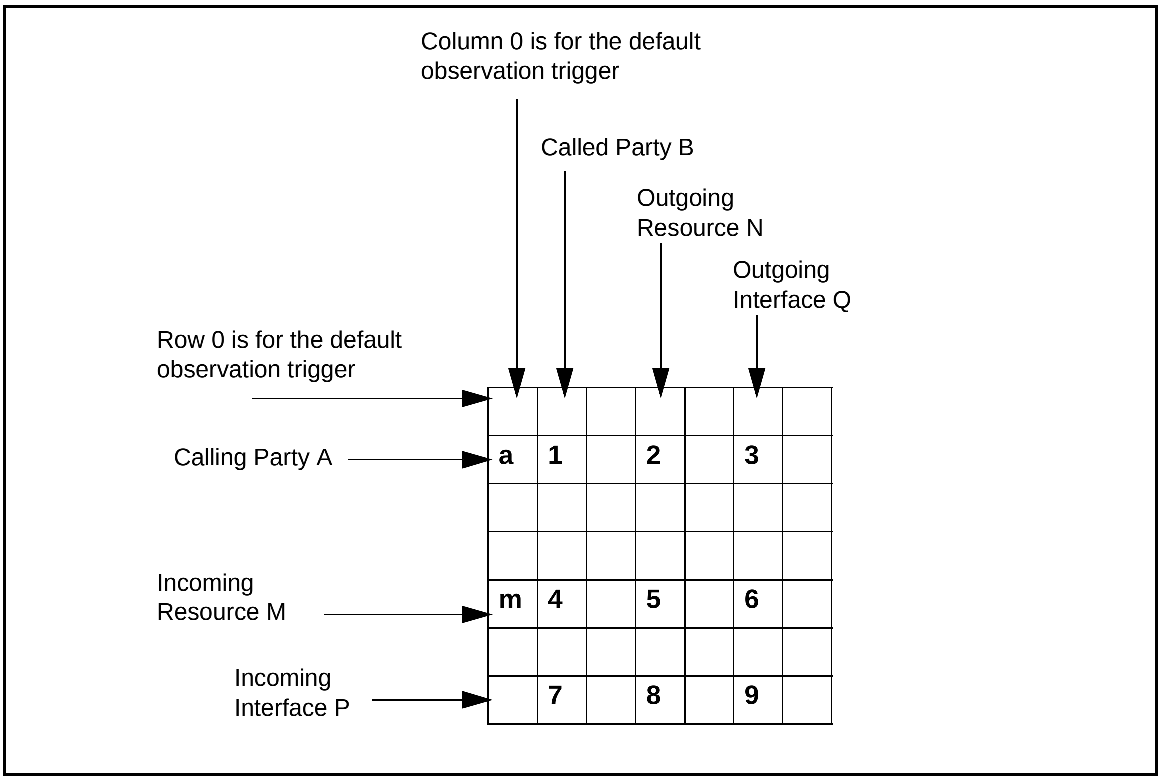


Figure 3-1: Observation trigger table

If an observation is desired only on a pair of calling party numbers and a called party number, the layer manager must ensure that, at the point where the row and column intersect, an appropriate observation trigger is set.

Refer to Figure 3-1 for an illustrated scenario of the columns and rows of the observation trigger table. The layer manager decides which trigger combinations are, or are not, meaningful.

- For the call originating from calling party A, going to called party B, the trigger value is 1.
- For the call originating from calling party A and leaving on outgoing resource N, the trigger value is 2.
- For the call on incoming resource M, going to called party B, the trigger value is 4.
- For the call on incoming resource M and leaving on outgoing resource N, the trigger value is 5.
- For the call on incoming resource M and leaving on outgoing interface Q, the trigger value is 6.
- For the call on incoming interface P, going to called party B, the trigger value is 7.
- For the call on incoming interface P, going to outgoing resource N, the trigger value is 8.
- For the call on incoming interface P and leaving on outgoing interface Q, the trigger value is 9.
- For the call originating from calling party A, going to called party B, and leaving on outgoing resource N, the trigger value is (1 OR 2); whereby, OR is a binary or operator.
- For the call originating from calling party A, going to called party B, and leaving on outgoing resource N and outgoing interface Q, the trigger value is (1 OR 2 OR 3).
- For the call originating from calling party A, coming in on incoming resource M, going to called party B, and leaving on outgoing resource N, the trigger value is (1 OR 2 OR 4 OR 5).
- For the call originating from calling party number A and coming in on incoming resource M, the trigger value is (a OR m).

Note: *ICC only supports the OR operation on triggers. For example, a call originating on calling party A, going to called party B, and leaving on outgoing interface Q is represented by (1 OR 3). However, the combination can be interpreted as tuples {{calling party A, called party B} AND {calling party A, outgoing interface Q}}; therefore, the result should be (1 AND 3).*

We allow a simple combination of triggers, and if any kind of complex triggering is necessary, the layer manager must build it.

3.4.1.1 Special Consideration for Connecting Private Networks

Phase 1 trunking solutions interconnect the devices to form Virtual Private Networks (VPNs). The following sections describe these in more detail.

Virtual Private Networks

A VPN is formed by connecting a group of private networks, such as PBXs of a corporation's different locations, using a backbone network. These VPNs use a private numbering plan for routing within a VPN and a public numbering plan for routing outside the VPN.

Figure 3-2 illustrates the VPN connection.

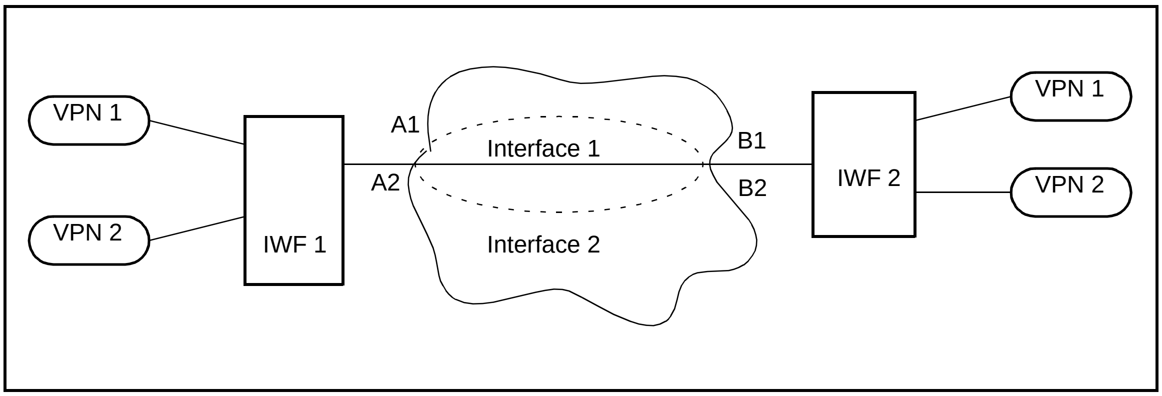


Figure 3-2: Virtual private networks

Two disjointed sets of VPNs may have identical numbering plans. The RT must identify the VPNs over which the call is originated at an IWF connecting to various VPNs (which avoid confusion between the identical numbering plans owned by the two VPNs). The information used by the RT to identify the VPNs over which the call is originated, is based on the interface, over which the call is originated. The following list contains information about this case.

- Each VPN must have one or more distinct logical interface(s) between the pair of IWFs.
- The RT uses the interface over which the incoming call is received to identify the VPN.
- For the private numbering plan-based called party numbers, the RT looks up the specific VPN in the routing tables.
- For the public numbering plan-based called party numbers, the RT performs a look-up in the global routing table.
- To define logical trunking interfaces per VPN, between a pair of IWFs, a tuple {calling party, called party} must be unique. This means that there must be a distinct calling party number for each VPN at the originating IWF. Since each of the two IWFs must originate and terminate, there is a distinct party number corresponding to each VPN.

  In Figure 3-2:

  - {A1, B1} identifies interface 1, and thus, VPN1 at both IWFs;
  - {A2, B2} identifies interface 2, and thus, VPN2 at both IWFs; and
  - A1 associates with VPN1 at IWF1 and A2 associates with VPN2 at IWF 1. B1 associates with VPN1 at IWF1 and B2 associates with VPN2 at IWF2.

3.4.1.2 Special Consideration for Trunking Interfaces

Figure 3-3 depicts the trunking interfaces.

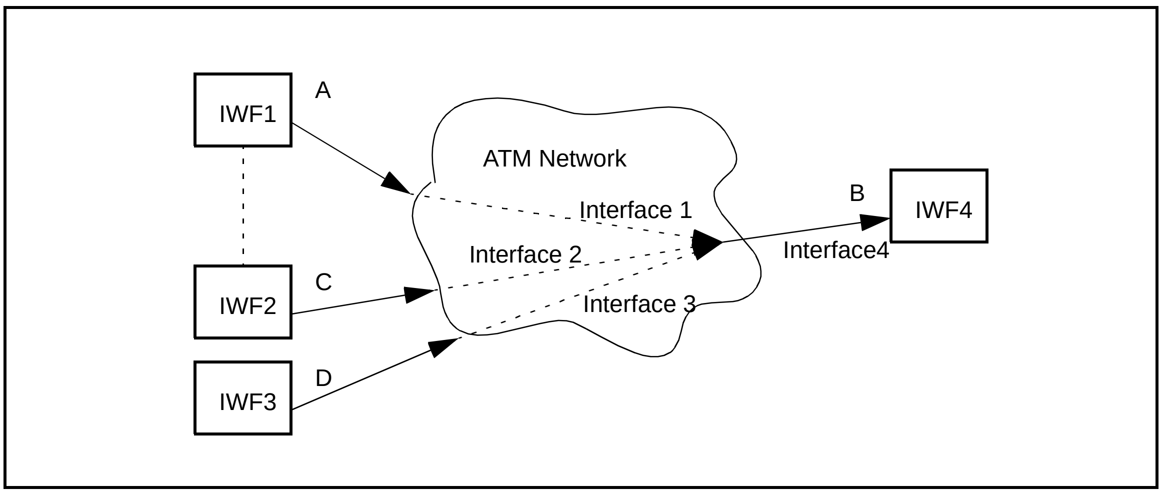


Figure 3-3: Trunking interfaces

For connecting multiple IWFs for the same VPNs, one called party B over interface 4 can be used to set up the trunking (signalling or bearer) connection with IWFs A, C, and D.

The RT at IWF4 must use a tuple {calling party, called party}, to extract the logical interface: {A, B} corresponding to interface 1.

3.4.1.3 Routing Tree

The VPN ID and some general information (address type, nature of address, and numbering plan) is added in front of the address before it is inserted into the routing tree. This additional information takes four digits—a three-digit routing address becomes seven digits when inserted into the routing tree. The worst case each route has as many nodes as the route digits, including the header that must be allocated.

Figure 3-4 illustrates a typical routing tree.

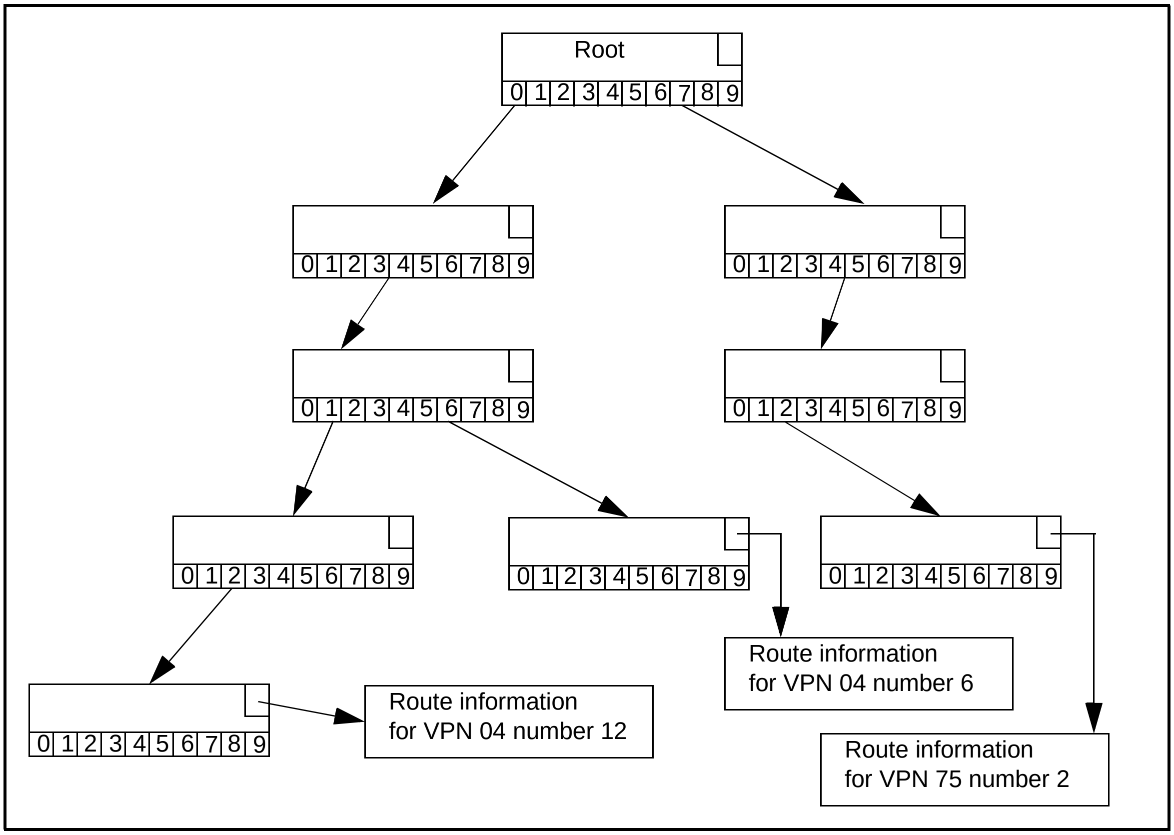


Figure 3-4: Routing tree

3.5 Generic Call Control

This section describes Generic Call Control (GCC), with in depth discussion of its interfaces and associated primitives.

3.5.1 Interface with the Layer Manager

This section discusses GCC's interface with its layer manager (LCC).

3.5.1.1 Primitive Overview

The following primitives are used between GCC and its layer manager.

Configuration

This procedure configures the protocol layer resources by using the following primitives.

Name	Description
`CcMiLccCfgReq`	Configuration request
`CcMiLccCfgCfm`	Configuration confirm

Control

This procedure activates and deactivates the protocol layer resources by using the following primitives.

Name	Description
`CcMiLccCntrlReq`	Control request
`CcMiLccCntrlCfm`	Control confirm

Statistics

This procedure retrieves statistics information by using the following primitives.

Name	Description
`CcMiLccStsReq`	Statistics request
`CcMiLccStsCfm`	Statistics confirm

Solicited Status

This procedure retrieves the status of GCC by using the following primitives.

Name	Description
`CcMiLccStaReq`	Status request
`CcMiLccStaCfm`	Status confirm

Unsolicited Status

This procedure indicates a status change of the protocol layer by using the following primitive.

Name	Description
`CcMiLccStaInd`	Status indication

Trace

The following primitive provides trace information to the layer manager.

Name	Description
`CcMiLccTrcInd`	Trace indication

Accounting

GCC provides Call Detail Record (CDR) information to the layer manager using the following primitive.

Name	Description
`CcMiLccAcntInd`	Accounting indication

3.5.1.2 Specific

This section describes in detail the primitives used between GCC and its layer manager.

3.5.1.2.1 CcMiLccCfgReq

Name:

Configuration Request

Direction:

Layer manager to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 CcMiLccCfgReq(pst, cfg)
Pst     *pst;
CcMngmt *cfg;
```

Parameters:

pst

For a description, see Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. The configuration structure has the following format:

```
typedef struct ccMngmt
{
   Header    hdr;               /* header */
   CmStatus  cfm;               /* status in confirm */
   union
   {      /* configuration */
       struct
      {
         union
         {
            CcGenCfg    ccGen;      /* Call Control General Config */
            CcPsSAPCfg  ccPSSap;    /* Protocol-Specific SAP Config */
            CcGenSAPCfg ccRMSap;    /* Resource Manager SAP Config */
            CcGenSAPCfg ccRTSap;    /* Router SAP Config */
            CcGenSAPCfg ccSFSap;    /* SF Manager SAP Config */
            CcIntfcCfg  ccIntfc;    /* Interface Configuration */
            CcBBProfCfg ccProf;     /* Broadband profile */
            CcObsTblCfg ccObsTblCfg;/* Observation trigger table
                                     * configuration
                                     */
            CcVIntfcCfg  ccVIntfc;  /* Virtual Interface Configuration */

          } s;
      } cfg;                        /* configuration */
   } t;
} CcMngmt;
```

hdr

Header structure. For more description, see Section 3.3.1, "Header."

cfm

Status field. For more information, see Section 3.3.2, "Status."

ccGen

General configuration structure. The general configuration must be done first. GCC uses much of the information carried by this table to reserve the proper amount of static memory.

```
typedef struct ccGenCfg         /* general configuration */
{
   U8 cid;                      /* Call Control Id  */
   U32 maxNmbCon;               /* Maximum Number of connections */
   U16 maxPsSAP;                /* Maximum number of PSIF SAPs */
   U16 maxRmSAP;                /* Maximum number of RM SAPs */
   U16 maxRtSAP;                /* Maximum number of RT SAPs */
   S16 timeRes;                 /* time resolution */
   CcGenTmrCfg tmr;             /* Call control general timers */
   Status poolTrLower;          /* Lower Threshold */
   Status poolTrUpper;          /* Upper Threshold */
   U16 maxSzeConHl;            /* Maximum size of SuConnId hash list */
   U16 maxSzeIntfcHl;          /* Maximum size of Interface hash list
                                 */
   U8 nmbProfId;                /* number of Broadband profiles */
   U8 minDgtsToRoute;          /* minimum digits required to initiate
                                   routing*/
   U8 countryCode[MAXCCODESIZE]; /* Country code associated with this
                                    call */
                                 /* control node */
   U8 trunkPrefix[MAXTPREFSIZE]; /* Trunk Prefix associated with this
                                    call */
                                 /* control node */
   PnNodeId ccNodeId;            /* Node Id of the call control - reqd
                                    for */
                                 /* PNNI routing */
   U8          obsTblDim;        /* Dimensions of an
                                  * observation trigger table
                                  */
   Bool        prfxCountryCode;  /* indicates whether country code is
                                    to be prefixed or not */
   U32         nmbIntfc;        /* Number of interfaces */
   U32         nmbTrnkdIntfc;   /* Number of trunked or
                                 * virtual interfaces
                                 */
   Pst sm;                      /* stack manager pst structure */
} CcGenCfg;
```

cid

GCC ID. An ID associates with each instance of GCC. It is used as the most significant octet of the **suConnId**, generated by GCC.
The allowable values are: 0 to 254.
The reserved value is: **CC_RESVD_CALL_CNTRL_ID.**

maxNmbCon

GCC requires this information to reserve the static memory required for GCC control blocks. The size of the hash list **icSuInstTbl** is also derived from this function and is proposed to be half the value of the **maxNmbCon** value. The size of the **icSpInstTbl** hash list maintained on each SAP is **maxNmbCon**, divided by the **maxPsSAPs** parameter.

maxPsSAP

Maximum number of the protocol-specific SAPs configured in GCC.

maxRmSAP

Maximum number of RM SAPs configured in GCC.

maxRtSAP

Maximum number of router SAPs configured in GCC. This includes both the static and PNNI router SAPs.

timeRes

Timer resolution, that is, the period during which the common timer function is called for this module. The module uses this period internally to maintain different timers for different connections.

tmr

For information on timer configuration, see Section 3.3.4, "Timer Configuration."

```
typedef struct ccGenTmrCfg
{
   TmrCfg tSETUP;     /* tSETUP timer - connection setup timer */
   TmrCfg tRLS;       /* tRLS timer - connection release timer */
   TmrCfg tCallDtl;   /* tCallDtl timer - Call Detail Info timer */
} CcGenTmrCfg;
```

tSETUP

This timer can be configured as a protective timer, which is started when GCC receives the **XxYyCctConInd** and is stopped after GCC initiates the **XxYyCctConRsp**, or after any other request to release the call is received. The expiration of this timer generally means that a primitive was lost (no route response, no resource allocation response, or no response from the SFM) or that there is another fault resulting in call clearing. The value of this timer should be larger than that of any protocol timer associated with the call setup.

`tRLS`

This timer can be configured to protect against primitive loss during the release phase of the call. This timer is started when the release procedure is initiated and is stopped only when the connection control block is deleted for the call. If this timer expires, an alarm is raised to the layer manager indicating the current state of the call for which this expiration has occurred. The connection control block is then released. Usually, the value of this timeout is in the order of minutes and should be sufficiently larger than that of any of the protocol timer values for release.

`tCallDtl`

An accounting indication, CDR, is generated when this timer expires, regardless of the fact that the call is still active.

`poolTrLower, poolTrUpper`

Upper and lower threshold levels for memory availability to GCC. If the system memory availability falls below any of these thresholds, no new calls are allowed.

`maxSzeConHl`

Size of the connection control block hash list. The ideal value is equal to the number of parallel systems existing in the system. In this case, each hash list bin has a maximum of one entry and the search time is minimal. By reducing the size of the hash list, the search time increases but the memory requirements are less. There is always a trade off between time and memory. A good value is about one fourth the number of connections, so that a hash list bin has a maximum of four entries.

`MaxSzeIntfcHl`

Size of the interface hash list. A good value is about one-fourth the number of interfaces specified in the **`nmbIntfc`**.

`nmbProfId`

Number of broadband profile tables. The broadband profile table contains the broadband information required to set up a broadband connection, which is not present when a call is originated from the narrowband side.

`minDgtsToRoute`

Number of digits that must be present in the called party number before GCC attempts routing.

countryCode

The country code forms a native E.164 format ATM endsystem address. This is required when a call is routed using the PNNI router and when the supplied E.164 address in the called party number is either a national or subscriber number, which must be converted to a native E.164 format AESA. **countryCode** is supplied as a NULL-terminated ASCII string.

trunkPrefix

The trunk prefix forms a native E.164 format ATM endsystem address. This is required when the call is routed using the PNNI router and when the supplied E.164 address in the called party number is a subscriber number, which must be converted to a native E.164 format AESA. **trunkPrefix** must be supplied as a NULL-terminated ASCII string.

ccNodeId

PNNI node ID. This is required when a call is routed using the PNNI router.

obsTblDim

Maximum number of rows and columns in the observation trigger table. Each row represents a criterion, based on the incoming parameters for an observation. Each column represents a criterion, based on the outgoing parameters for an observation. The value of this variable specifies the maximum number of incoming criteria or outgoing criteria set for observations. The allowable values are: 1 to **LCC_MAX_OBS_TBLSZ**.

prfxCountryCode

Prefix country code. This field indicates whether the country code must be prefixed in the calling party number received. If it is set to TRUE, the country code will be prefixed to the calling party number if it is not already present.

nmbIntfc

This specifies the total number of interfaces supported in the system. These can be regular ISDN, ISUP, PNNI interfaces, or the virtual interfaces created for the trunking or feature transparency solution.

nmbTrnkdIntfc

This is the total number of virtual interfaces configured in the system. These are virtual interfaces created for trunking or feature transparency solutions. For configuring these interface types, use the **STCCVINTFC** element type.

sm

Post structure. It is used for communicating with the stack manager. GCC uses the post structure when sending unsolicited status, which is sent to the address in the **sm** field.

ccPSSap

Protocol-Specific Interface Function (PSIF) SAP configuration. This SAP communicates with the incoming/outgoing protocols.

```
typedef struct ccPsSAPCfg       /* PSIF Sap Configuration structure */
{
   SuId suId;                   /* service user id to be configured */
   SpId spId;                   /* service provider id */
   S16 sapType;                 /* sap type */
   Ent dstEnt;                  /* entity */
   Inst dstInst;                /* instance */
   ProcId dstProcId;            /* destination processor id */
   Priority prior;              /* priority */
   Route route;                 /* route */
   Selector selector;           /* selector */
   MemoryId mem;                /* memory region & pool id */
   CcSapTmrCfg tmr;             /* SAP timers */
} CcPsSAPCfg;
```

suId

Service user ID. GCC uses this information to identify the SAP.

spId

Service provider ID. GCC passes this to the PSIF for all interactions. The PSIF uses **spId** to identify the SAP on which it communicates with GCC.

sapType

Identifies the protocol type and its variant used by the underlying PSIF. This field is not used currently.

dstEnt, **dstInst**

Destination entity ID and the destination process instance ID associated with this SAP. This has significance only for loosely coupled entities.

dstProcId

Processor ID of the processor on which the destination entity resides.

prior

Priority used when the task buffers must be posted between the service provider and service user. It is used only in a loosely coupled system.

route

The system uses this for internal routing requirements. TAPA does not define the contents or use of this information. It is used only in a loosely coupled system.

selector

Defines whether the service provider and service user are loosely or tightly coupled. It is used to resolve a primitive call to the lower layer (PSIF).

mem

For more information, refer to Section 3.3.5, "**Memory**."

tmr

For information on timer configuration, see Section 3.3.4, "Timer Configuration."

```
typedef struct ccSapTmrCfg
{
   TmrCfg tINTERDGT;      /* tINTERDGT timer - inter digit timer */
   TmrCfg tBNDCFM;        /* tBNDCFM timer - Bind Confirm timer */
   TmrCfg t25ISUP;
   TmrCfg t26ISUP;
   TmrCfg t37ISUP;
} CcSapTmrCfg;
```

tINTERDGT

Indicates the value of the inter-digits time-out run to detect the end of the called party number. This timer can be configured if the incoming signalling type supports overlap signalling. The value contained in this parameter indicates the time (expressed in seconds), which elapses after the last digit is received, after which it is assumed that more digits are not expected for this call.

tBNDCFM

When GCC binds its lower SAP to PSIF's upper SAP, GCC sends a bind request to the PSIF. The PSIF responds with the bind confirm when the necessary processing is done. When it sends a bind request, GCC starts the timer **tBNDCFM**. When the timer expires, GCC retries binding the PSIF by sending another bind request. GCC tries for a fixed number of times (**CC_MAX_RETRY** defined in **cc.h**) before declaring that the bind procedure failed. If the bind procedure fails, GCC sends an alarm (**LCM_EVENT_BND_FAIL**).

t25ISUP

This timer must be configured when the **sapType** is a variant of ISUP. The value for this timer should be in accordance with the range allowed by the protocol specification for that particular variant.

t26ISUP

This timer must be configured when the **sapType** is a variant of ISUP. The value for this timer should be in accordance with the range allowed by the protocol specification for that particular variant.

t37ISUP

This timer must be configured when the **sapType** is a variant of ISUP. The value for this timer should be in accordance with the range allowed by the protocol specification for that particular variant.

ccRMSap, ccRTSap, ccSFSap

These parameters contain the configuration for the RT (**ccRTSap**), RM (**ccRMSap**), and SFM (**ccSFSap**). For all these SAPs, the same information is necessary and the same structure is used:

```
typedef struct ccGenSAPCfg     /* General Sap Configuration structure */
{
   SuId suId;                  /* service user id to be configured */
   SpId spId;                  /* service provider id */
   U8 sapType;                 /* sap type */
   Ent dstEnt;                 /* entity */
   Inst dstInst;               /* instance */
   ProcId dstProcId;           /* destination processor id */
   Priority prior;             /* priority */
   Route route;                /* route */
   Selector selector;          /* selector */
   MemoryId mem;               /* memory region & pool id */
   CcGenSapTmrCfg tmr;         /* SAP timers */
} CcGenSAPCfg;
```

suId

Service user ID. GCC requires this information to identify the SAP.

spId

Service provider ID. GCC passes this to the PSIF for all interactions. The PSIF uses this **spId** to identify the SAP on which it communicates with GCC.

sapType

This information is currently used only for the RT SAP and it specifies the router (dynamic and static) type to which this SAP is bound. GCC uses this information to set up the route request according to the router used. The allowable values are:

```
CC_STATIC_ROUTER      1
CC_PNNI_ROUTER        2
```

dstEnt, dstInst

Destination entity ID and the destination process instance ID associated with this SAP. These fields have significance only for loosely coupled entities.

dstProcId

Processor ID of the processor on which the destination entity resides.

prior

Priority used when the task buffers must be posted between the service provider and service user. It is used only in a loosely coupled system.

route

The system uses this for internal routing requirements. TAPA does not define the contents or use of this information. It is used only in a loosely coupled system.

selector

Defines whether the service provider and service user are loosely or tightly coupled. It is used to resolve a primitive call to the lower layer (PSIF).

mem

For more details, refer to Section 3.3.5, "**Memory**."

tmr

Timer configuration. For more information, see Section 3.3.4, "Timer Configuration."

```
typedef struct ccGenSapTmrCfg
{
   TmrCfg tBNDCFM; /* tBNDCFM timer - Bind Confirm timer */
#ifdef ICC_AUDIT
   TmrCfg tAUDCFM; /* tAUDCFM timer - timer to wait on SftAudCfm */
#endif /* ICC_AUDIT */
} CcGenSapTmrCfg;
```

tBNDCFM

When GCC binds its lower SAP to the PSIF's upper SAP, it sends a bind request to the PSIF. The PSIF responds with the bind confirm when the necessary processing is done. When it sends a bind request, GCC starts the timer **tBNDCFM**. When the timer expires, GCC retries binding the PSIF by sending another bind request. GCC tries for a fixed number of times (**CC_MAX_RETRY**, defined in **cc.h**) before declaring that the bind procedure has failed. If the bind procedure fails, GCC sends an alarm (**LCM_EVENT_BND_FAIL**).

tAUDCFM

In the RM auditing procedure, GCC sends an audit request toward the SF to disconnect all the connections, which are associated with the **suConnId**s passed in the audit request. Upon sending this audit request, GCC starts the **tAUDCFM** timer. When the timer expires at the first time before receiving an audit confirm from the SF, GCC re-sends the audit request to the SF. When the timer expires at the second time, GCC sends an alarm with the cause **LCM_CAUSE_TMR_EXPIRED**.

ccIntfc

It is used to configure the interfaces in the system.

```
typedef struct ccIntfcCfg   /* Interface SAP Configuration structure */
{
   RmInterface intfc;           /* Interface */
   U8 destSAPid;                /* Destination SAP Id */
   U8 destRMSAPid;              /* Destination RMSAP Id */
   U8 numRTSAPs;                /* number of the associated RTSAPs */
   U8 destRTSAPId[MAXRTSAP];    /* Identification of the associated
                                   RTSAPs */
} CcIntfcCfg;
```

intfc

Interface for which the destination SAP information is configured. For more information, see Section 3.3.6, "Interface."

destSAPid

Destination SAP ID communicating with the PSIF associated with the identified interface.

destRMSAPid

The RM SAP ID allocating resources for this interface. This identification allows for having different RMs in the system.

numRTSAPs

The number of RT SAPs configured for this interface—two RT SAPs is the maximum that can be configured.

destRTSAPid

List of router SAP IDs used to route the calls originating at this interface. Each interface can have a number of different routers (up to **MAXRTSAP**) configured per incoming interface (For example, a PNNI router and static router).

MAXRTSAP 2

To route a call, call control selects in order the router SAPs configured at the incoming interface associated with the call. For interworking scenarios involving the PNNI interface as one of the originating or terminating interfaces, both PNNI and the static router are required for routing the call.

When the PNNI router is used, it is configured as the first SAP and the static router is configured as the second SAP in the array of SAP IDs, for the DSS2 interfaces (PNNI, Q.93B). For non-PNNI interfaces (ISUP, B-ISUP), the static router is configured as the first SAP and the PNNI router is configured as the second SAP in the array of the router SAPs.

ccProf

Broadband profile configuration. This table contains information about using the Constant Bit Rate (CBR) and Variable Bit Rate (VBR) services, and about introducing in B-ISUP messages certain parameters that do not have correspondents in ISUP's original messages (for example, broadband bearer capability). Configure this table before configuring the narrowband circuits. This configuration is required only for interworking.

```
typedef struct ccBBProfCfg      /* Broadband Profile Configuration
                                   Structure */
{
   U8 profId;                   /* profile identifier */
   U8 profType;                 /* profile type */
   union
   {
      CcAtmParms  ituProf;      /* ITU BB profile */
      AalConParam atmProf;      /* ATM BB profile */
   }t;
} CcBBProfCfg;
```

profId

Profile ID. The allowable values are: 0 to 255.

profType

Profile type. The allowable value is **CC_ITU_PROFILE.**

ituProf

ITU ATM parameter profile. GCC allows configuring only the **ituProf**. For details, refer to Section APPENDIX A:, "Broadband Profile."

atmProf

It is not used in GCC.

ccObsTblCfg

This stores an observation trigger including signalling conversion analysis. Each row corresponds to a trigger based on the incoming parameters (such as calling party number, incoming resource) and each column corresponds to the outgoing parameters (such as called party number, outgoing resource).

```
typedef struct ccObsTblCfg
{
   U8   obsType;
   union
   {
      CcObsTblElmntArray elmntArray;
      CcObsTblElmnt      elmnt;
   }r;
} CcObsTblCfg;
```

obsType

This specifies whether a row or column is configured.

The allowable values are:

Value	Description
LCC_OBS_ROW	Configures a row in the observation trigger table
LCC_OBS_COL	Configures a column in the observation trigger table
LCC_OBS_ELMNT	Configures a column in the observation trigger table

elmntArray

This structure is used to set a row or column in the observation table. The structure **ccObsTblElmntArray** is used when the **obsType** is **LCC_OBS_ROW** or **LCC_OBS_COL**.

```
typedef struct ccObsTblElmntArray
{
   U8   obsIdx;
   U8   numEnt;
   U8   entry[LCC_MAX_OBS_TBLSZ];
} CcObsTblElmntArray;
```

obsIdx

This specifies the trigger row or column that the layer manager wants to either create or over write.

numEnt

This specifies the number of columns and/or rows that have been filled corresponding to the **obsIdx**.

entry

Each element in the entry has flags set to indicate whether a signalling conversion analysis must be triggered for a particular **obsIdx**.

The allowable values are a combination of these bitmasks:

Value	Description
LCC_TRIG_STATMC	Trigger state transition
LCC_TRIG_CCT_LOG	Trigger protocol events
LCC_TRIG_MSG_DUMP	Trigger protocol events with event dumps
LCC_TRIG_RMT_LOG	Trigger RMT events
LCC_TRIG_RTT_LOG	Trigger RTT events
LCC_TRIG_SFT_LOG	Trigger SFT events
LCC_TRIG_TMR_LOG	Trigger Timer events

elmnt

This structure is used to set a row or column of the observation table. The structure **ccObsTblElmnt** is used when the **obsType** is **LCC_OBS_ELMNT**.

```
typedef struct ccObsTblElmnt
{
   U8   row;
   U8   col;
   U8   val;
} CcObsTblElmnt;
```

row

This specifies the row of the element to be modified.

col

This specifies the column of the element to be modified.

val

This value set for the specified entry.

The allowable values are a combination of these bitmasks:

Values	Description
LCC_TRIG_STATMC	Trigger state transition
LCC_TRIG_CCT_LOG	Trigger protocol events
LCC_TRIG_MSG_DUMP	Trigger protocol events with event dumps
LCC_TRIG_RMT_LOG	Trigger RMT events
LCC_TRIG_RTT_LOG	Trigger RTT events
LCC_TRIG_SFT_LOG	Trigger SFT events
LCC_TRIG_TMR_LOG	Trigger timer events

ccVIntc

This configures virtual interfaces. Virtual interfaces are used for the Trunking / Tunneling Call Control (TCC). A virtual interface is defined between a pair of nodes that may not be adjacent to each other in the network, in that they are not directly connected by a set of physical links. Virtual interfaces are defined between a pair of IWFs.

```
typedef struct ccVIntfcCfg      /* Virtual Interface SAP Configuration
                                 * structure
                                 */
{
   RmInterface trnkdIntfc;      /* Trunked Interface */
   U8 trnkdPsSapId;             /* Trunked PSIF SAP Id */
   U8 trnkdRMSapId;             /* Trunked RMSAP Id */
   U8 numTrnkdRTSaps;           /* number of the RTSAPs associated
                                 * with trunked interface
                                 */
   U8 trnkdRtSapId[MAXRTSAP];   /* Identification of the RTSAPs
                                   associated
                                 * with trunked interface
                                 */
   U8 trnkgIntfType;            /* Identifies the Interface type */
   U8 trnkgPsSapId;             /* Trunking PSIF SAP Id */
   U8 trnkgRMSapId;             /* Trunking RMSAP Id */
   U8 numTrnkgRTSaps;           /* number of the RTSAPs associated
                                 * with Trunking interface
                                 */
   U8 trnkgRtSapId[MAXRTSAP];   /* Identification of the RTSAPs
                                   associated
                                 * with Trunking interface
                                 */
} CcVIntfcCfg;
```

TrnkdIntfc

Interface for which the destination SAP information is configured. For more information, refer to Section 3.3.6, "Interface."

The allowable **intfcType**s are:

CC_SI_INTFC
CC_IN_INTFC

trnkdPsSapId

Destination SAP ID communicating with the PSIF associated with the identified interface.

trnkdRMSapId

The RM SAP ID that allocates resources for this interface. This identification allows for having different RMs in the system.

numTrnkdRTsaps

Number of RT SAPs configured for this interface. The maximum that can be configured is two.

trnkdRtSapId

List of router SAP IDs used to route the calls originating at this interface. Each interface can have a number of different routers (up to **MAXRTSAP**) configured per incoming interface (For example, a PNNI router and static router).

MAXRTSAP 2

To route a call, call control selects in order the router SAPs configured at the incoming interface associated with the call. For interworking scenarios involving the PNNI interface as one of the originating or terminating interfaces, both PNNI and the static router are required to route the call. When the PNNI router is used, it is configured as the first SAP and the static router is configured as the second SAP, in the array of SAP IDs for the DSS2 interfaces (PNNI, Q.93B).

For non-PNNI interfaces (ISUP, B-ISUP), the static router is configured as the first SAP and the PNNI router is configured as the second SAP in the array of the router SAPs.

TrnkgIntfc

Interface for which the destination SAP information is configured. For more details, see Section 3.3.6, "Interface."

The allowable **intfcType**s are:

CC_FEATTRP_INTFC
CC_PH1TK_INTFC

trnkgPsSapId

Destination SAP ID communicating with the PSIF associated with the identified interface.

trnkgRMSapId

RM SAP ID that allocates resources for this interface. This identification allows for having different RMs in the system.

numTrnkgRTsaps

Number of RT SAPs configured for this interface. The maximum that can be configured is two.

trnkgRtSapId

List of router SAP IDs used to route the calls originating at this interface. Each interface can have a number of different routers (up to **MAXRTSAP**) configured per incoming interface (For example, a PNNI router and static router).

MAXRTSAP 2

To route a call, call control selects in order the router SAPS configured at the incoming interface associated with the call. For interworking scenarios involving the PNNI interface as one of the originating or terminating interfaces, both PNNI and the static router are required to route the call. When the PNNI router is used, it is configured as the first SAP and the static router is configured as the second SAP, in the array of SAP IDs for the DSS2 interfaces (PNNI, Q.93B).

For non-PNNI interfaces (ISUP, B-ISUP), the static router is configured as the first SAP and the PNNI router is configured as the second SAP in the array of the router SAPs.

Description:

The layer manager uses this function to configure GCC.

Returns:

00 ROK

01 RFAILED

3.5.1.2.2 CcMiLccCfgCfm

Name:

Configuration Confirm

Direction:

GCC to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 CcMiLccCfgCfm(pst, cfg)
Pst     *pst;
CcMngmt *cfg;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. With the exception of the following fields, the structure used for the configuration confirm is the same as that for the configuration request. For more information, see Section 3.5.1.2.1, "**CcMiLccCfgReq**."

cfm

The status field has the following format.

```
typedef struct cmStatus
{
    U16   status;              /* Status of the operation  */
    U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous configuration request primitive. It contains one of the following values.

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it.

Name	Description
LCM_REASON_MEM_NOAVAIL	Memory allocation failed
LCM_REASON_REGTMR_FAIL	**SRegTmr** returned failure
LCM_REASON_GENCFG_NOT_DONE	Configuration request received without previous general configuration
LCM_REASON_EXCEED_CONF_VAL	Maximum value as given in the general configuration is exceeded. For example, the layer manager tries to configure SAP 5, however, the maximum number of SAPs passed in the general configuration is 4.
LCM_REASON_RECONFIG_FAIL	Failure in reconfiguration
LCM_REASON_INVALID_SAP	Invalid SAP value passed. The passed SAP does not exist in the system.
LCM_REASON_HASHING_FAILED	Hash list library returned failure
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header

Note: *The remaining fields are the same as those passed in the configuration request. For more information on this, see* Section 3.5.1.2.1, "**CcMiLccCfgReq**."

3.5.1.2.3 CcMiLccCntrlReq

Name:

Control Request

Direction:

Layer manager to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 CcMiLccCntrlReq(pst, cntrl)
Pst     *pst;
CcMngmt *cntrl;
```

Parameters:

pst

For more details, see Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. The control structure has the following format.

```
typedef struct ccMngmt
{
   Header hdr;                  /* header */
   CmStatus  cfm;               /* status in confirm */
   union
   {
/* control */
   struct
      {
         DateTime dt;                 /* date and time */
         U8 type;                     /* type of control */
         U8 action;                   /* action */
         union
         {
            U32         dbgMask;      /* debug mask */
            UConnId     suConnId;     /* Connection Identifier */
            RmInterface  intfc;       /* Interface */
            CcTestCallCntrl testCall; /* test call control */
            SuId         suId;        /* SAP id */
            ProcId       dstProcId;   /* Destination procId for group
                                       * actions */
         } c;
      } cntrl;                        /* control */
```

hdr

For more details, see Section 3.3.1, "Header."

cfm

It is used only in confirm primitives.

dt

Date and time structure.

type/subAction

Type of control procedure requested. The allowable values are:

Name	Description
SAUSTA	Unsolicited status generation
SAELMNT	Specified element
SAGR_DSTPROCID	Specified group elements
LCC_STKSTRT	Stack start command
SADBG	Debug information generation
LCC_CLEARCONN	Clear a connection
SATRC	Trace generation
SAACNT	Accounting information generation

action

Specific action code. The allowable values are:

Name	Description
AENA	Enable
ADISIMM	Disable immediately

dbgMask

Bit mask of different debug classes enabled or disabled. This specifies the classes of debug messages that must be controlled (enabled or disabled). The following debug class is defined.

Name	Description
DBGMASK_CC	Internal GCC debug class

suConnId

Connection ID used to identify a connection. **suConnId** is used for the control request to clear a connection.

intfc

Interface to be controlled. For more information, refer to Section 3.3.6, "Interface."

testCall

Information to initiate the test call.

```
typedef struct ccTestCallCntrl
{
   RmRsc        resource;
   Action       actionDetail;
} CcTestCallCntrl;
```

resource

The resource that initiates the test call. Refer to **typedef struct rmRsc /* Generic Resource Structure */** on page 197 for more details.

actionDetail

Action initiated during a test call. The allowable value is:

LCC_CONTINUITY_CALL	**Continuity test call**

suId

The service user ID to identify the lower SAP of GCC.

dstProcId

The processor ID to identify the group of lower SAPs of GCC, toward the same processor.

Description:

This function controls the GCC layer. The following table contains the possible operations with the required parameters.

Description	**type**	**action**	**Others**
Enables the alarms	**SAUSTA**	**AENA**	**N/A**
Disables the alarms		**ADISIMM**	
Starts the stack	**LCC_STKSTRT**	N / A	
Enables a debug class	**SADBG**	**AENA**	**dbgMask**
Disables a debug class		**ADISIMM**	**dbgMask**
Clears a connection	**LCC_CLEARCONN**	N / A	**suConnId**
Enables a CDR generation	**SAACNT**	**AENA**	**hdr.elmId.elmnt** = **STCCCDR.** The **intfc** at which the CDR is enabled.
Disables a CDR generation		**ADISIMM**	**hdr.elmId.elmnt** = **STCCCDR.** The **intfc** at which the CDR is enabled.
Enables the signalling conversion analysis	**SATRC**	**AENA**	**hdr.elmId.elmnt** = **STCCOBS**
Disables the signalling conversion analysis		**ADISIMM**	**hdr.elmId.elmnt** = **STCCOBS**
Enables the PS SAP	**SAELMNT**	**AENA**	**hdr.elmId.elmnt = STCCPSSAP** and the **cntrl.c.suId** is an SAP ID of the PS SAP.
Bind and enable PS SAP		**ABND_ENA**	**hdr.elmId.elmnt = STCCPSSAP** and the **cntrl.c.suId** is an SAP ID of the PS SAP.
Disables the PS SAP gracefully		**ADISGRC**	**hdr.elmId.elmnt = STCCPSSAP** and the **cntrl.c.suId** is an SAP ID of the PS SAP.
Disable the immediate PS SAP		**ADISIMM**	**hdr.elmId.elmnt = STCCPSSAP** and the **cntrl.c.suId** is an SAP ID of the PS SAP.

Description	**type**	**action**	**Others**
Unbind and disable PS SAP		**AUBND_DIS**	**hdr.elmId.elmnt = STCCPSSAP** and the **cntrl.c.suId** is an SAP ID of the PS SAP.
Delete the PS SAP		**ADEL**	**hdr.elmId.elmnt = STCCPSSAP** and the **cntrl.c.suId** is an SAP ID of the PS SAP.
Enable the RM SAP		**AENA**	**hdr.elmId.elmnt = STCCRMSAP** and the **cntrl.c.suId** is an SAP ID of the RM SAP.
Bind and enable RM SAP		**ABND_ENA**	**hdr.elmId.elmnt = STCCRMSAP** and the **cntrl.c.suId** is an SAP ID of the RM SAP.
Disable the RM SAP gracefully		**ADISGRC**	**hdr.elmId.elmnt = STCCRMSAP** and the **cntrl.c.suId** is an SAP ID of the RM SAP.
Disable the immediate RM SAP		**ADISIMM**	**hdr.elmId.elmnt = STCCRMSAP** and the **cntrl.c.suId** is an SAP ID of the RM SAP.
Unbind and disable RM SAP		**AUBND_DIS**	**hdr.elmId.elmnt = STCCRMSAP** and the **cntrl.c.suId** is an SAP ID of the RM SAP.
Delete the RM SAP		**ADEL**	**hdr.elmId.elmnt = STCCRMSAP** and the **cntrl.c.suId** is an SAP ID of the RM SAP.
Enable the RT SAP		**AENA**	**hdr.elmId.elmnt = STCCRTSAP** and the **cntrl.c.suId** is an SAP ID of the RT SAP.
Bind and enable RT SAP		**ABND_ENA**	**hdr.elmId.elmnt = STCCRTSAP** and the **cntrl.c.suId** is an SAP ID of the RT SAP.

Description	**type**	**action**	**Others**
Disable the RT SAP gracefully		**ADISGRC**	**hdr.elmId.elmnt = STCCRTSAP** and the **cntrl.c.suId** is an SAP ID of the RT SAP.
Disable the immediate RT SAP		**ADISIMM**	**hdr.elmId.elmnt = STCCRTSAP** and the **cntrl.c.suId** is an SAP ID of the RT SAP.
Unbind and disable RT SAP		**AUBND_DIS**	**hdr.elmId.elmnt = STCCRTSAP** and the **cntrl.c.suId** is an SAP ID of the RT SAP.
Delete the RT SAP		**ADEL**	**hdr.elmId.elmnt = STCCRTSAP** and the **cntrl.c.suId** is an SAP ID of the RT SAP.
Enable the SF SAP		**AENA**	**hdr.elmId.elmnt = STCCSFSAP** and the **cntrl.c.suId** is an SAP ID of the SF SAP.
Bind and enable SF SAP		**ABND_ENA**	**hdr.elmId.elmnt = STCCSFSAP** and the **cntrl.c.suId** is an SAP ID of the SF SAP.
Disable the SF SAP gracefully		**ADISGRC**	**hdr.elmId.elmnt = STCCSFSAP** and the **cntrl.c.suId** is an SAP ID of the SF SAP.
Disable the immediate SF SAP		**ADISIMM**	**hdr.elmId.elmnt = STCCSFSAP** and the **cntrl.c.suId** is an SAP ID of the SF SAP.
Unbind and disable SF SAP		**AUBND_DIS**	**hdr.elmId.elmnt = STCCSFSAP** and the **cntrl.c.suId** is an SAP ID of the SF SAP.
Delete the SF SAP		**ADEL**	**hdr.elmId.elmnt = STCCSFSAP** and the **cntrl.c.suId** is an SAP ID of the SF SAP.
Enables an interface		**AENA**	**hdr.elmId.elmnt** = **STCCINTFC.** The **intfc** enables.

Description	**type**	**action**	**Others**
Disables an interface		**ADISIMM**	**hdr.elmId.elmnt** = **STCCINTFC.** The **intfc** disables.
Deletes an interface		**ADEL**	**hdr.elmId.elmnt** = **STCCINTFC.** The **intfc** deletes.
Enables a virtual interface		**AENA**	**hdr.elmId.elmnt** = **STCCINTFC** and **CcTestCallCntrl.**
Disables a virtual interface		**ADISIMM**	**hdr.elmId.elmnt** = **STCCINTFC.** The virtual interface disables.
Deletes a virtual interface		**ADEL**	**hdr.elmId.elmnt** = **STCCINTFC.** The virtual interface deletes.
Initiates a test call		**AENA**	**hdr.elmId.elmnt** = **STCCTSTCALL.** The virtual interface deletes.
Shut down the GCC		**ASHUTDOWN**	**hdr.elmId.elmnt = STGEN.**
Enable a group of **PsSaps.**	**SAGR_DSTPROCID**	**AENA**	**hdr.elmId.elmnt = STCCGRPSSAP.**
Enable a group of **RmSaps.**		**AENA**	**hdr.elmId.elmnt = STCCGRRMSAP.**
Enable a group of **RtSaps**		**AENA**	**hdr.elmId.elmnt = STCCGRRTSAP.**
Enable a group of **SfSaps**		**AENA**	**hdr.elmId.elmnt = STCCGRSFSAP.**
Disable a group of **PsSaps** immediately		**AUBND_DIS**	**hdr.elmId.elmnt = STCCGRPSSAP.**
Disable a group of **RmSaps** immediately		**AUBND_DIS**	**hdr.elmId.elmnt = STCCGRRMSAP.**
Disable a group of **RtSaps** immediately		**AUBND_DIS**	**hdr.elmId.elmnt = STCCGRRTSAP.**
Disable a group of **SfSaps** immediately		**AUBND_DIS**	**hdr.elmId.elmnt = STCCGRSFSAP.**

Description	type	action	Others
Delete a group of **PsSaps**		**ADEL**	**hdr.elmId.elmnt = STCCGRPSSAP.**
Delete a group of **RmSaps**		**ADEL**	**hdr.elmId.elmnt = STCCGRRMSAP.**
Delete a group of **RtSaps**		**ADEL**	**hdr.elmId.elmnt = STCCGRRTSAP.**
Delete a group of **SfSaps**		**ADEL**	**hdr.elmId.elmnt = STCCGRSFSAP.**

Note: *Enabling a signalling conversion analysis does so for all calls, however, GCC generates any signalling conversion information only if a trigger is installed, and if one of the calls matches the criterion specified in any of the triggers.*

Returns:

00 **ROK**

01 **RFAILED**

3.5.1.2.4 CcMiLccCntrlCfm

Name:

Control Confirm

Direction:

GCC to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 CcMiLccCntrlCfm(pst, cntrl)
Pst     *pst;
CcMngmt *cntrl;
```

Parameters:

pst

For more details, refer to Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. With the exception of the following fields, the structure used for control confirm is the same as that for the control request. For more information, see Section 3.4.1.2.3, "**CcMiLccCntrlReq**."

cfm

The status field indicates the result of a request. The status field has the following format.

```
typedef struct cmStatus
{
    U16   status;             /* Status of the operation  */
    U16   reason;             /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values.

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The remaining fields are the same as those passed in the control request. For more details, see Section 3.5.1.2.3.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_INVALID_ACTION	Invalid action passed in the control structure
LCM_REASON_INVALID_SUBACTION	Invalid subaction passed in the control structure
LCC_REASON_INVALID_RMSAP	Invalid RM SAP
LCC_REASON_INVALID_RTSAP	Invalid RT SAP
LCC_REASON_INVALID_SFSAP	Invalid SFM SAP

3.5.1.2.5 CcMiLccStsReq

Name:

Statistics Request

Direction:

Layer manager to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 CcMiLccStsReq(pst, action, sts)
Pst     *pst;
Action  action;
CcMngmt *sts;
```

Parameters:

pst

For more description, refer to Section 3.3.3, "**Pst**."

action

Action indicator. The allowable values are:

Value	Description
0	Zero statistics counters (**ZEROSTS**)
1	Do not set the statistics counters to zero (**NOZEROSTS**)

sts

Pointer to the statistics structure. It has the following format:

```
typedef struct ccMngmt
{
   Header hdr;
   union
   {
/* statistics */
   struct
      {
         DateTime dt;                /* date and time */
         Duration dura;              /* duration */
         union
         {
            CcGenSts   ccGenSts;     /* General statistics */
           CcPsSAPSts  ccPsSapSts;    /* Protocol-Specific SAP statistics */
            CcIntfcSts ccIntfcSts;   /* Interface statistics */
         } s;
      } sts;                         /* statistics */
   } t;
} CcMngmt;
```

hdr

For more description, see Section 3.3.1, "Header."

The type of statistics information desired can be selected by programming the header substructure as:

elmnt

Element. The allowable values are:

Value	Description
STGEN	General
STSAP	SAP statistics
STCCINTFC	Interface-specific statistics
STCCVINTFC	Virtual interface-specific statistics

dt

Date and time structure.

dura

Duration structure.

ccGenSts

General statistic counters. This field is not relevant to the statistics request. **CcMiLccStsCfm** returns the values. For more information, see Section 3.5.1.2.6, "**CcMiLccStsCfm**."

ccPsSapSts

The PSIF SAP statistic counters. This field is not relevant to the statistics request. The statistics confirm primitive returns the values. The only field that must be set is **suId**, which specifies the SAP for which the statistics are requested. For more information, refer to Section 3.5.1.2.6, "**CcMiLccStsCfm**."

```
typedef  struct ccPsSAPSts      /* PSIF SAP statistics */
{
   SuId  suId;                  /* SuId of the associated SAP */
   Cntr  incoming;              /* Number of incoming calls */
   Cntr  outgoing;              /* Number of outgoing calls */
} CcPsSAPSts;
```

suId

Number of the SAP for which statistics are requested. This field must be filled when the layer manager sends the status request to GCC.

incoming, outgoing

These fields have relevance only to the confirm primitive. For information, see Section 3.5.1.2.6, "**CcMiLccStsCfm**."

ccIntfcSts

Interface statistic counters. This field is not relevant to the statistics request. The statistics confirm primitive returns the values. The only field that must be set is **intfc**, which specifies the interface for which statistics are requested. The other fields are explained in the statistics confirm primitive. For details, see Section 3.5.1.2.6, "**CcMiLccStsCfm**."

```
typedef struct ccIntfcSts       /* Interface statistics */
{
   RmInterface intfc;           /* Interface */
   StsCntr  numIcCallAttempt;   /* Nbr of incoming call attempt */
   StsCntr  numIcCallAnswered;  /* Nbr of incoming call answered */
   StsCntr  numOgCallAttempt;   /* Nbr of Outgoing call attempt */
   StsCntr  numOgCallAnswered;  /* Nbr of Outgoing call answered */
} CcIntfcSts;
```

`itfc`

Interface for which statistics are requested. This field must be filled when the layer manager sends the status request to GCC.

`numIcCallAttempt, numIcCallAnswered, numOgCallAttempt, numOgCallAnswered`

These fields have relevance only to the confirm primitive. For more information, see Section 3.5.1.2.6, "**`CcMiLccStsCfm`**."

Description:

The layer manager uses this function to gather statistics information.

Returns:

00 ROK

01 RFAILED

3.5.1.2.6 CcMiLccStsCfm

Name:

Statistics Confirm

Direction:

GCC to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 CcMiLccStsCfm(pst, sts)
Pst     *pst;
CcMngmt *sts;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sts

Pointer to the statistics structure. The statistics structure has the following format.

```
typedef struct ccMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
/* statistics */
   struct
      {
         DateTime dt;               /* date and time */
         Duration dura;             /* duration */
         union
         {
            CcGenSts    ccGenSts;   /* General statistics */
            CcPsSAPSts  ccPsSapSts; /* Protocol Specific SAP statistics */
            CcIntfcSts  ccIntfcSts; /* Interface statistics */
         } s;
      } sts;                        /* statistics */
   } t;
} CcMngmt;
```

hdr

Header structure. For further details, refer to Section 3.3.1, "Header."

cfm

Status field. The status field indicates the result of a request. It has the following format.

```
typedef struct cmStatus
{
   U16  status;              /* Status of the operation  */
   U16  reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values.

Name	Description
LCM_PRIM_OK	Statistics request successful
LCM_PRIM_NOK	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCC_REASON_INVALID_SAP	Invalid SAP specified

dt

Date and time structure.

dura

Duration structure.

ccGenSts

General statistic counters. This structure is defined as:

```
typedef struct ccGenSts     /* General statistics */
{
   Cntr totalCalls;         /* Total Number of ConnId Handled */
   Cntr answered;           /* Total Number of Calls Answered */
   Cntr fRoutUnavail;       /* Calls failed-no route to Called party */
   Cntr fResUnavail;        /* Calls failed due to resource
                               unavailability */
} CcGenSts;
```

totalCalls

Number of calls, successful and unsuccessful, handled in GCC. GCC fills this field, and it is returned via the **CcMiLccStsCfm**.

answered

Number of successful calls, which are those that reached the answered state. GCC fills this field, and it is returned via the **CcMiLccStsCfm**.

fRoutUnavail

Number of calls that failed because a route was not available. GCC fills this field and is returned via the **CcMiLccStsCfm**.

fResUnavail

Number of calls that failed due to resource failure. GCC fills this field and is returned via the **CcMiLccStsCfm**.

ccPsSapSts

PSIF SAP statistic counters.

```
typedef struct ccPsSAPSts       /* PSIF SAP statistics */
{
   SuId suId;                   /* SuId of the associated SAP */
   Cntr  incoming;              /* Number of incoming calls */
   Cntr  outgoing;              /* Number of outgoing calls */
} CcPsSAPSts;
```

suId

Number of the SAP for which statistics are requested. The layer manager fills this field when sending the status request to GCC.

incoming

Number of incoming calls, both successful and unsuccessful.

outgoing

Number of outgoing calls, both successful and unsuccessful.

ccIntfcSts

Interface statistic counters. The layer manager fills this field when sending the status request to GCC.

```
typedef struct ccIntfcSts       /* Interface statistics */
{
   RmInterface intfc;           /* Interface */
   StsCntr  numIcCallAttempt;   /* Number of incoming call attempt */
   StsCntr  numIcCallAnswered;  /* Number of incoming call answered */
   StsCntr  numOgCallAttempt;   /* Number of Outgoing call attempt */
   StsCntr  numOgCallAnswered;  /* Number of Outgoing call answered */
} CcIntfcSts;
```

itfc

Interface for which statistics are requested. The layer manager fills this field when sending the status request to GCC.

numIcCallAttempt

Number of incoming call attempts.

numIcCallAnswered

Number of incoming calls that reached the answered state.

numOgCallAttempt

Number of outgoing call attempts.

numOgCallAnswered

Number of outgoing calls that reached the answered state.

Description:

GCC uses this function to provide the layer manager with statistics information.

Returns:

00 ROK

01 RFAILED

3.5.1.2.7 CcMiLccStaReq

Name:

Status Request

Direction:

Layer manager to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 CcMiLccStaReq(pst, sta)
Pst      *pst;
CcMngmt  *sta;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sta

Pointer to the solicited status structure. The solicited status structure has the following format.

```
typedef struct ccMngmt
{
   Header hdr;
   CmStatus  cfm;               /* status in confirm */
   union
   {
 /* solicited status */
      struct
      {
         DateTime dt;           /* date and time */
         union
         {
            SystemId    sysId;      /* System Id */
            CcIntfcSta  ccIntfc;    /* Interface Status */
            CcObsTblSta ccObsTbl;   /* Observation trigger table
            CcSapSta    ccSap;      /* Sap status */

         } s;
      } ssta;                   /* solicited status */
   } t;
} CcMngmt;
```

hdr

Header structure. For a description, see Section 3.3.1, "Header."

The type of status information desired can be selected by programming the header substructure as:

elmnt

Element. The allowable values are:

Value	Description
STSID	System ID
STCCINTFC	Interface-specific status
STCCVINTFC	Virtual interface-specific status
STCCOBS	Observation trigger table status
STPSSAP	PS SAP status
STRMUPSAP	RM SAP status
STRTSAP	RT SAP status
STSFSAP	SF SAP status

cfm

Status field. Only the confirmation primitives use this field to report errors. It is not significant to the status request. For further information, see Section 3.3.2, "Status."

dt

Date and time structure. It is previously described.

sysId

System ID for GCC. The status confirm primitive returns the values.

ccIntfc

Interface status. The status confirm primitive returns the values. Only the **intfc** field must be set, which specifies the interface for which status is requested. The other fields are explained in the statistics confirm primitive. For more information, see Section 3.5.1.2.8, "**CcMiLccStaCfm**."

```
typedef struct ccIntfcSta       /* Interface SAP Status structure */
{
   RmInterface intfc;           /* Interface */
   U8 destSAPid;                /* Destination SAP Id */
   U8 destRMSAPid;              /* Destination RMSAP Id */
   U8 numRTSAPs;                /* number of the associated RTSAPs */
   U8 destRTSAPId[MAXRTSAP];    /* Identification of the associated
                                   RTSAPs */
   U16 nmbActvConn;             /* Number of active connections on
                                 * the interface
                                 */
   State state;                 /* Interface control block state */
} CcIntfcSta;
```

intfc

Interface for which status is requested. This field must be filled when the layer manager sends the status request to GCC.

destRMSAPid, numRTSAPs, destRTSAPId[MAXRTSAP], nmbActvConn, and **state**

These fields are relevant only to the confirm primitive. For more details, refer to Section 3.5.1.2.8, "**CcMiLccStaCfm**."

ccObsTbl

Observation table status. The status confirm primitive returns the values. The fields that must be set are **obsType.** If **obsType** is set to **LCC_OBS_ELMNT elmnt.row**, then **elmnt.col** must be specified.

```
typedef struct ccObsTblCfg
{
   U8   obsType;
   union
   {
      CcObsTblElmntArray elmntArray;
      CcObsTblElmnt      elmnt;
   }r;
} CcObsTblCfg;
```

obsType

Specifies whether the status of a row or column is requested. This field must be filled when the layer manager sends the status request to GCC.

The possible values are:

Value	Description
LCC_OBS_ROW	Configures a row in the observation trigger table
LCC_OBS_COL	Configures a column in the observation trigger table
LCC_OBS_ELMNT	Configures a column in the observation trigger table

elmntArray

This structure is used to return a row or column of the observation table. The structure **ccObsTblElmntArray** is used when the the status is requested for **obsType LCC_OBS_ROW** or **LCC_OBS_COL**.

```
typedef struct ccObsTblElmntArray
{
   U8   obsIdx;
   U8   numEnt;
   U8   entry[LCC_MAX_OBS_TBLSZ];
} CcObsTblElmntArray;
```

obsIdx

Specifies the trigger row or column for which status is requested. This field must be filled when the layer manager sends the status request to GCC.

numEnt, entry

These fields are relevant only to the confirm primitive. For further details, refer to Section 3.5.1.2.8, "**CcMiLccStaCfm**."

elmnt

This structure is used to return one value in the observation table. The structure **ccObsTblElmnt** is used when the **obsType** is **LCC_OBS_ELMNT**.

```
typedef struct ccObsTblElmnt
{
   U8   row;
   U8   col;
   U8   val;
} CcObsTblElmnt;
```

row

Specifies the row. This field must be filled when the layer manager sends the status request to GCC.

col

Specifies the column. This field must be filled when the layer manager sends the status request to GCC.

val

These fields are relevant only to the confirm primitive. For more details, see Section 3.5.1.2.8, "**CcMiLccStaCfm**."

ccSap

Sap Status for any of the lower saps (PS, RM, RT and SF).

```
typedef struct ccSapSta
{
    SuId suId;
    SpId spId;
    S16  state;
    S16  sapType;
    S8   bndRetryCount;
} CcSapSta;
```

suId

Service user ID of the SAP.

spId

Service provider ID.

state

State of the SAP—bound, unbound, or binding in progress.

sapType

Type of SAP. This field is valid only for the RT SAP. The allowed values of the RT SAP type are **CC_PNNI_ROUTER** and **CC_STATIC_ROUTER.**

Description:

The layer manager uses this function to gather solicited status information.

Returns:

00 **ROK**

01 **RFAILED**

3.5.1.2.8 CcMiLccStaCfm

Name:

Status Confirm

Direction:

GCC to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 CcMiLccStaCfm(pst, sta)
Pst     *pst;
CcMngmt *sta;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sta

Pointer to the solicited status structure. The solicited status structure has the following format.

```
typedef struct ccMngmt
{
   Header hdr;
   CmStatus  cfm;              /* status in confirm */
   union
   {
 /* solicited status */
      struct
      {
         DateTime dt;          /* date and time */
         union
         {
            SystemId    sysId;      /* System Id */
            CcIntfcSta  ccIntfc;    /* Interface Status */
            CcObsTblSta ccObsTbl;   /* Observation trigger table
                                     * Status
                                     */
            CcSapSta    ccSap;      /* Sap Status */
         } s;
      } ssta;                  /* solicited status */
   } t;
} CcMngmt;
```

hdr

Header structure. For more description, refer to Section 3.3.1, "Header."

cfm

The status field indicates the result of a request. The status field has the following format.

```
typedef struct cmStatus
{
   U16   status;              /* Status of the operation  */
   U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values.

Name	Description
LCM_PRIM_OK	Statistics request successful
LCM_PRIM_NOK	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header

dt

Date and time structure.

sysId

System ID for ICC.

ccIntfc

Interface status.

```
typedef struct ccIntfcSta       /* Interface SAP Status structure */
{
   RmInterface intfc;           /* Interface */
   U8 destSAPid;                /* Destination SAP Id */
   U8 destRMSAPid;              /* Destination RMSAP Id */
   U8 numRTSAPs;                /* number of the associated RTSAPs */
   U8 destRTSAPId[MAXRTSAP];    /* Identification of the associated
                                 * RTSAPs
                                 */
   U16 nmbActvConn;             /* Number of active connections on
                                 * the interface
                                 */
   State state;                 /* Interface control block state */
} CcIntfcSta;
```

intfc

Interface for which status is requested. This field must be filled when the layer manager sends the status request to GCC.

destSAPid

Destination SAP ID that communicates with the PSIF associated with the identified interface.

destRMSAPid

The RM SAP ID that allocates resources for this interface. This identification allows for having different RMs in the system.

numRTSAPs

Number of RT SAPs configured for this interface.

destRTSAPid

List of router SAP IDs used to route the calls originating at this interface. Each interface can have a number of different routers (For example, a PNNI router and a static router). For more information, refer to Section 3.5.1.2.1, "**CcMiLccCfgReq**."

ccObsTbl

Observation table status.

```
typedef struct ccObsTblCfg
{
   U8   obsType;
   union
   {
      CcObsTblElmntArray elmntArray;
      CcObsTblElmnt      elmnt;
   }r;
} CcObsTblCfg;

typedef CcObsTblCfg CcObsTblSta;
```

obsType

Specifies whether the status of a row or a column is requested.

The possible values are:

Value	Description
LCC_OBS_ROW	Configures a row in the observation trigger table
LCC_OBS_COL	Configures a column in the observation trigger table
LCC_OBS_ELMNT	Configures a column in the observation trigger table

elmntArray

This structure is used to return a row or column of the observation table. The structure **ccObsTblElmntArray** is used when the status is requested for the **obsType** **LCC_OBS_ROW** or **LCC_OBS_COL**.

```
typedef struct ccObsTblElmntArray
{
   U8   obsIdx;
   U8   numEnt;
   U8   entry[LCC_MAX_OBS_TBLSZ];
} CcObsTblElmntArray;
```

obsIdx

This specifies the trigger row or column for which status is requested. This field must be filled when the layer manager sends the status request to GCC.

numEnt

This specifies the number of columns or rows corresponding to this **obsIdx**.

entry

Each element in the entry has flags set to indicate whether a signalling conversion analysis must be triggered for a particular **obsIdx**.

The possible values are a combination of these bitmasks:

Value	Description
LCC_TRIG_STATMC	Triggers the state transition
LCC_TRIG_CCT_LOG	Triggers the protocol events
LCC_TRIG_MSG_DUMP	Triggers the protocol events with event dumps
LCC_TRIG_RMT_LOG	Triggers the RMT events
LCC_TRIG_RTT_LOG	Triggers the RTT events
LCC_TRIG_SFT_LOG	Triggers the SFT events
LCC_TRIG_TMR_LOG	Triggers the timer events

elmnt

This structure is used to return one value in the observation table. The struct **ccObsTblElmnt** is used when the **obsType** is **LCC_OBS_ELMNT**.

```
typedef struct ccObsTblElmnt
{
   U8   row;
   U8   col;
   U8   val;
} CcObsTblElmnt;
```

row

This specifies the row. This field must be filled when the layer manager sends the status request to GCC.

col

Specifies the column. This field must be filled when the layer manager sends the status request to GCC.

val

The value set for the specified entry. The possible values are a combination of these bitmasks:

Value	Description
LCC_TRIG_STATMC	Triggers the state transition
LCC_TRIG_CCT_LOG	Triggers the protocol events
LCC_TRIG_MSG_DUMP	Triggers the protocol events with event dumps
LCC_TRIG_RMT_LOG	Triggers the RMT events
LCC_TRIG_RTT_LOG	Triggers the RTT events
LCC_TRIG_SFT_LOG	Triggers the SFT events
LCC_TRIG_TMR_LOG	Triggers the timer events

ccSap

SAP status for any of the lower SAPs—PS, RM, RT, and SF.

```
typedef struct ccSapSta
{
    SuId suId;
    SpId spId;
    S16  state;
    S16  sapType;
    S8   bndRetryCount;
} CcSapSta;
```

suId

Service user ID of the SAP.

spId

Service provider ID.

state

State of the SAP—bound, unbound, or binding in progress.

sapType

Type of SAP. This field is valid only for the RT SAP. The allowed values of the RT SAP type are **CC_PNNI_ROUTER** and **CC_STATIC_ROUTER.**

Description:

GCC uses this function to return the solicited status information to the layer manager.

Returns:

00 ROK

01 RFAILED

3.5.1.2.9 CcMiLccStaInd

Name:

Status Indication

Direction:

GCC to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 CcMiLccStaInd(pst, sta)
Pst     *pst;
CcMngmt *sta;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sta

Pointer to the unsolicited status structure. The status structure has the following format.

```
typedef struct ccMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
/* unsolicited status */
      struct
      {
         CmAlarm  alarm;              /* alarm */
         union
         {
            SpId         spId;        /* service provider id */
            UConnId      suConnId;    /* service user instance id */
            RmInterface  intfc;       /* Interface */
         }t;
      } usta;                         /* unsolicited status */
   } t;
} CcMngmt;
```

hdr

Header structure. For more description, refer to Section 3.3.1, "Header."

cfm

The status field is not significant in this primitive.

alarm

Alarm. It contains the category, event, and cause of the alarm. The descriptions of these fields follow this structure format:

```
typedef struct cmAlarm

{
   DateTime dt;       /* data and time */
   U16 category;      /* alarm category*/
   U16 event;         /* alarm event */
   U16 cause;         /* alarm cause */
}CmAlarm;
```

dt

Date and time structure.

category

This field describes the category to which the error belongs.

Name	Description
LCM_CATEGORY_PROTOCOL	Protocol error. This can occur while mapping one message from the incoming to the outgoing side.
LCM_CATEGORY_RESOURCE	GCC cannot allocate memory.
LCM_CATEGORY_INTERFACE	When an event is received on a SAP that is not configured or bound
LCM_CATEGORY_INTERNAL	Internal errors, such as hash list failures

Note: *All the categories do not apply for all the options.*

event

This field specifies the event that has occurred.

Name	Description
LCM_EVENT_INV_STATE	Event received in the invalid state
LCM_EVENT_LYR_SPECIFIC	Protocol layer-specific mapping error
LCM_EVENT_LI_INV_EVT	Invalid event received from the lower layer (CCT interface)
LCC_EVENT_HASHING_FAILED	Hash list error
LCC_EVENT_INV_DESTSAP	Invalid destination SAP
LCC_EVENT_INV_RMSAP	Invalid RM SAP
LCC_EVENT_INV_RT	Invalid RT SAP
LCC_EVENT_INV_SF	Invalid SFM SAP
LCC_EVENT_MAPPING_FAILED	Mapping library function returned failure

Note: *All the event values are not valid for all the options.*

cause

This field specifies the cause. Additional information in union **t** depends on the cause.

Name	Description
LCM_CAUSE_INV_SAP	The invalid SAP. The value causing the problem is passed in the **spId** field.
LCC_CAUSE_INV_INTERFACE	The invalid interface is specified. The interface value causing the problem is passed in the interface field.
LCC_CAUSE_MALLOC_FAIL	Memory could not be allocated. The SAP on which the problem occurred is passed in the **spId** field.
LCC_CAUSE_SUINSTTBL_INS	The value that could not be located is passed in the **val** field.
LCC_CAUSE_SPINSTTBL_INS	The value that could not be located is passed in the **val** field.
LCC_CAUSE_SUINSTTBL_FIND	The value that could not be located is passed in the **val** field.
LCC_CAUSE_SPINSTTBL_FIND	The value that could not be located is passed in the **val** field.

Name	Description
LCC_CAUSE_INTFCSAPTBL_FIND	This interface could not be located in the hash list; it is passed in the interface field.
LCC_CAUSE_MAPFAIL_NBBB	The mapping from ISUP to B-ISUP failed. Additional information is passed.
LCC_CAUSE_MAPFAIL_BBNB	The mapping from ISUP broadband to ISUP narrowband failed. Additional information is not passed.
LCC_CAUSE_MAPFAIL_NBNB	The mapping from ISUP to ISUP failed. Additional information is not passed.
LCC_CAUSE_MAPFAIL_BBBB	The mapping from B-ISUP to B-ISUP failed. Additional information is not passed.
LCC_CAUSE_RELEASETMR_EXP	GCC has cleared the connection control block, although the release confirm has not been received. Expiration of timer **tRLC.**
LCC_CAUSE_SETUPTMR_EXP	The connection has been released because the setup timer expired.
LCC_CAUSE_MAPFAIL_ININ	The mapping from ISDN to ISDN failed. Additional information is not passed.
LCC_CAUSE_MAPFAIL_SIIN	The mapping from ISUP to ISDN failed. Additional information is not passed.
LCC_CAUSE_MAPFAIL_INSI	The mapping from ISDN to ISUP failed. Additional information is not passed.
LCC_CAUSE_PSSAPBNDTMREXP	GCC did not receive a bind confirmation from the PSIF.
LCC_CAUSE_RMSAPBNDTMREXP	GCC did not receive a bind confirmation from the RM.
LCC_CAUSE_RTSAPBNDTMREXP	GCC did not receive a bind confirmation from the RT.
LCC_CAUSE_SFSAPBNDTMREXP	GCC did not receive a bind confirmation from the SFM.
LCC_CAUSE_MAPFAIL_AMAM	The mapping from Q.93B to Q.93B failed. Additional information is not passed.
LCC_CAUSE_MAPFAIL_AMSI	The mapping from Q.93B to ISUP failed. Additional information is not passed.
LCC_CAUSE_MAPFAIL_SIAM	The mapping from ISUP to Q.93B failed. Additional information is not passed.
LCC_CAUSE_DEALOC_IND	GCC received a resource deallocation indication from the RM.

Note: *All the cause values are not valid for all the options.*

spId

The SAP ID associated with the **LCM_CAUSE_INV_SAP** alarm.

suConnId

The connection ID associated with the alarms: **LCC_CAUSE_SUINSTTBL_INS**, **LCC_CAUSE_SPINSTTBL_INS**, **LCC_CAUSE_SUINSTTBL_FIND**, and **LCC_CAUSE_SPINSTTBL_FIND**.

intfc

The interface associated with **LCC_CAUSE_INTFCSAPTBL_FIND** and **LCC_CAUSE_INV_INTERFACE** alarms.

Description:

GCC uses this function to provide the layer manager with unsolicited status information (alarms). The unsolicited status can be enabled or disabled via the layer manager control request.

Description	Category	Event	Cause
The GCC primitives received the invalid SAP ID.	**LCM_CATEGORY_INTERFACE**	**LCM_EVENT_UI_INV_EVT**	**LCM_CAUSE_INV_SAP**
The bind confirmation from the PSIF indicates failure.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_BND_FAIL**	**LCM_CAUSE_UNKNOWN**
The SAP state associated with the GCC primitive is not bound.	**LCM_CATEGORY_INTERFACE**	**LCM_EVENT_INV_STATE**	**LCM_CAUSE_INV_SAP**
Memory could not be allocated to store the incoming connection event in the connection control block, because static memory was unavailable.	**LCM_CATEGORY_RESOURCE**	**LCC_EVENT_MEMALOC_FAILED**	**LCC_CAUSE_MALLOC_FAIL**
The connection control block could not be inserted in the **suConnId** hash list.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_HASHING_FAILED**	**LCC_CAUSE_SUINSTTBL_INS**
The connection control block could not be inserted in the **spConnId** hash list.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_HASHING_FAILED**	**LCC_CAUSE_SPINSTTBL_INS**
The connection control block could not be found in the **suConnId** hash list.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_HASHING_FAILED**	**LCC_CAUSE_SUINSTTBL_FIND**
The connection control block could not be found in the **suConnId** hash list or the **spConnId** hash list.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_HASHING_FAILED**	**LCC_CAUSE_SPINSTTBL_FIND**

Description	Category	Event	Cause
The connection control block could not be found in the interface SAP hash list.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_HASHING_FAILED**	**LCC_CAUSE_INTFCSAPTBL_FIND**
The destination PSIF SAP associated with the **CcLiCctMntStaInd** is not bound.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_INV_DESTSAP**	**LCM_CAUSE_INV_SAP**
The destination RT SAP associated with the **CcLiCctMntStaInd** is not bound.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_INV_RTSAP**	**LCM_CAUSE_INV_SAP**
The following RM primitives received the invalid SAP ID.	**LCM_CATEGORY_INTERFACE**	**LCC_EVENT_INV_RMSAP**	**LCM_CAUSE_INV_SAP**
The bind confirmation from the RM indicates failure.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_BND_FAIL**	**LCM_CAUSE_UNKNOWN**
The RT primitives received the invalid SAP ID.	**LCM_CATEGORY_INTERFACE**	**LCC_EVENT_INV_RTSAP**	**LCM_CAUSE_INV_SAP**
The bind confirmation from the RT indicates failure.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_BND_FAIL**	**LCM_CAUSE_UNKNOWN**
The SFM primitives received the invalid SAP ID.	**LCM_CATEGORY_INTERFACE**	**LCC_EVENT_INV_RTSAP**	**LCM_CAUSE_INV_SAP**
The bind confirmation from the SFM indicates failure.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_BND_FAIL**	**LCM_CAUSE_UNKNOWN**
The connection setup timer expired.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_INV_STATE**	**LCC_CAUSE_SETUPTMR_EXP**
The mapping library function returned failure.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_INV_STATE**	**LCC_CAUSE_RELEASETMR_EXP**
The connection control block could not be found in the **suConnId** hash list during a layer manager-initiated connection.	**LCM_CATEGORY_INTERNAL**	**LCC_EVENT_HASHING_FAILED**	**LCC_CAUSE_SUINSTTBL_FIND**
The bind confirmation timer expired in call control for the PSIF SAP bind request, and the maximum bind retry count has been reached.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_BND_FAIL**	**LCCC_CAUSE_PSSAPBNDTMREXP**
The bind confirmation timer expired in call control for the RM SAP bind request, and the maximum bind retry count has been reached.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_BND_FAIL**	**LCCC_CAUSE_RMSAPBNDTMREXP**

Description	Category	Event	Cause
The bind confirmation timer expired in call control for the RT SAP bind request, and the maximum bind retry count has been reached.	**LCM_CATEGORY_ PROTOCOL**	**LCM_EVENT_BND _FAIL**	**LCCC_CAUSE_RT SAPBNDTMREXP**
The bind confirmation timer expired in call control for the SF SAP bind request, and the maximum bind retry count has been reached.	**LCM_CATEGORY_ PROTOCOL**	**LCM_EVENT_BND _FAIL**	**LCCC_CAUSE_SF SAPBNDTMREXP**
The destination RM SAP associated with the incoming interface provided in the **CcLiCctConInd** is invalid.	**LCM_CATEGORY_ INTERNAL**	**LCC_EVENT_INV _RMSAP**	**LCM_CAUSE_INV _SAP**
The destination RT SAP associated with the incoming interface provided in the **CcLiCctConInd** is invalid.	**LCM_CATEGORY_ INTERNAL**	**LCC_EVENT_INV _RTSAP**	**LCM_CAUSE_INV _SAP**
The destination SF SAP associated with the incoming interface provided in the **CcLiCctConInd** is invalid.	**LCM_CATEGORY_ INTERNAL**	**LCC_EVENT_INV _SFSAP**	**LCM_CAUSE_INV _SAP**
The incoming interface provided in the **CcLiCctConInd** could not be found in the interface SAP hash list.	**LCM_CATEGORY_ INTERNAL**	**LCC_EVENT_HAS HING_FAILED**	**LCC_CAUSE_INT FCSAPTBL_FIND**
The connection control block could not be allocated due to static memory unavailability.	**LCM_CATEGORY_ RESOURCE**		**LCC_CAUSE_MAL LOC_FAIL**
ISUP-to-B-ISUP mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_ PROTOCOL**	**LCC_EVENT_MAP PING_FAILED**	**LCC_CAUSE_MAP FAIL_NBBB**
B-ISUP-to-ISUP mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_ PROTOCOL**	**LCC_EVENT_MAP PING_FAILED**	**LCC_CAUSE_MAP FAIL_BBNB**
ISUP-to-ISUP mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_ PROTOCOL**	**LCC_EVENT_MAP PING_FAILED**	**LCC_CAUSE_MAP FAIL_NBNB**

Description	Category	Event	Cause
B-ISUP-to-B-ISUP mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_PROTOCOL**	**LCC_EVENT_MAPPING_FAILED**	**LCC_CAUSE_MAPFAIL_BBBB**
ISUP-to-B-ISUP mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_PROTOCOL**	**LCC_EVENT_MAPPING_FAILED**	**LCC_CAUSE_MAPFAIL_BBNB**
ISDN-to-ISDN mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_PROTOCOL**	**LCC_EVENT_MAPPING_FAILED**	**LCC_CAUSE_MAPFAIL_ININ**
ISDN-to-ISUP mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_PROTOCOL**	**LCC_EVENT_MAPPING_FAILED**	**LCC_CAUSE_MAPFAIL_INSI**
ISUP-to-ISDN mapping failed. The index identifying the failed mapping function is supplied with the alarm.	**LCM_CATEGORY_PROTOCOL**	**LCC_EVENT_MAPPING_FAILED**	**LCC_CAUSE_MAPFAIL_SIIN**
ISUP-to-B-ISUP mapping failed because an unsupported information element was received in the event to be mapped. The identity of the unsupported information element is supplied with the alarm.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_LYR_SPECIFIC**	**LCC_CAUSE_MAPFAIL_NBBB**
B-ISUP-to-ISUP mapping failed because an unsupported information element was received in the event to be mapped. The identity of the unsupported information element is supplied with the alarm.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_LYR_SPECIFIC**	**LCC_CAUSE_MAPFAIL_BBNB**
Memory could not be allocated for the buffer required to hold unrecognized information.	**LCM_CATEGORY_RESOURCE**	**LCC_EVENT_MEMALOC_FAILED**	**LCC_CAUSE_MALLOC_FAIL**
The RM received the resource deallocation indication.	**LCM_CATEGORY_PROTOCOL**	**LCM_EVENT_INV_STATE**	**LCC_CAUSE_DEALOC_IND**

Returns:

00 ROK

01 RFAILED

3.5.1.2.10 CcMiLccTrcInd

Name:

Trace Indication

Direction:

GCC to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 CcMiLccTrcInd(pst, trc)
Pst     *pst;
CcMngmt *trc;
```

Parameters:

pst

For more description, refer to Section 3.3.3, "**Pst**."

trc

Pointer to the trace indication structure. The trace structure has the following format.

```
typedef struct ccMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
/* Trace Indication */
      struct
      {
         DateTime dt;                /* date and time */
         U16      evnt;              /* event */
         union
         {
#ifdef CC_GEN_OBS_TRC
            CcObsTrc   ccObsTrc; /* Observation trace information */
#endif /* CC_GEN_OBS_TRC */
             CcNullElmt  nl;     /* Null element for compilation */
         }u;
      } trc;                         /* Trace indication */
   } t;
} CcMngmt;
```

hdr

Header structure. For more details, see Section 3.3.1, "Header."

cfm

The status field is not significant in this primitive.

dt

Date and time structure.

evnt

The following field specifies the event that has occurred:

Name	**Description**
LCC_OBS_TRC	Signalling conversion analysis trace

ccObsTrc

This structure is used to report the trace information corresponding to the observations triggers that are configured.

```
typedef struct ccObsTrc
{
   SuId       icSuId;          /* incoming sap Id */
   SuId       ogSuId;          /* outgoing sap Id */
   UConnId    icSuConnId;      /* Incoming SuConnId */
   UConnId    ogSuConnId;      /* Outgoing SuConnId */
   U8         icProtType;      /* incoming protocol type */
   U8         ogProtType;      /* outgoing protocol type */
   U8         routerType;      /* Router Type */
   RmRsc      icRsc;           /* Incoming Resource */
   RmRsc      ogRsc;           /* Outgoing Resource */
   U8         obsType;         /* Obervation type */
   struct
   {
      U8      icRscIdx;     /* incoming resource observation index */
      U8      icIntfcIdx;   /* incoming interface observation index */
      U8      cgPtyIdx;     /* calling party number observation index*/
      U8      ogRscIdx;     /* outgoing resource observation index */
      U8      ogIntfcIdx;   /* outgoing interface observation index */
      U8      cdPtyIdx;     /* called party number observation index */
   }obsIdx;
   U8         evntType;        /* State machine event type */
   U8         subEvntType;     /* State machine sub-event type */
   SuId       sapId;           /* sapId on which the event is
                                * received.
                                */
   UConnId    spConnId;        /* connection identifier on
                                * which the event is received.
                                */
   State      crntCallState;   /* current call state */
   U8         evntPres;        /* Detailed event present */
   CcAllObsTrcEvnts *evnt;     /* Pointer to the event
                                * structure
                                */
} CcObsTrc;
```

icSuId

SAP ID of the incoming PSIF SAP over which the call is received.

ogSuId

SAP ID of the outgoing PSIF SAP to which the call is sent.

icSuConnId

Connection handle of the incoming half of the call.

ogSuConnId

Connection handle of the outgoing half of the call.

icProtType

Protocol type of the incoming PSIF SAP. For further details, refer to Section 3.3.7, "Protocol Variants."

ogProtType

Protocol type of the outgoing PSIF SAP. For further details, refer to Section 3.3.7, "Protocol Variants."

routerType

The router used to route the call, for example, the static router or PNNI routing.

icRsc

Incoming resource.

ogRsc

Outgoing resource.

obsType

Observation type. The allowable values are:

Value	Description
LCC_CCT_EVNT	Event, received from / sent to, the protocol SAP
LCC_RMT_EVNT	Event, received from / sent to, the RMT interface
LCC_RTT_EVNT	Event, received from / sent to, the RTT interface
LCC_SFT_EVNT	Event, received from / sent to, the SFT interface
LCC_TMR_EVNT	Event, received from / sent to, the timer
LCC_STMC_EVNT	State machine transitions

icRscIdx

This is the row index in the observation table corresponding to the incoming resource.

icIntfcIdx

This the row index in the observation table corresponding to the incoming interface.

cgPtyIdx

This is the row index in the observation table corresponding to the called party number.

ogRscIdx

This is the column index in the observation table corresponding to the outgoing resource.

ogIntfcIdx

This is the column index in the observation table corresponding to the outgoing interface.

cdPtyIdx

This is the row index in the observation table corresponding to the called party number.

Note: *If the indexes are unknown, they are zero. When the index becomes available, it is replaced with the received value.*

evntType

State machine event type. The **evntType** corresponds to the primitives sent or received by GCC, such as **CCE_CONIND**.

subEvntType

In case one primitive maps to multiple events, **subEvntType** provides additional information. For example, **CcLiCctCnStInd** has an **evntType** as a parameter, which is mapped to the **subEvntType**.

sapId

SAP ID on which the event is received.

spConnId

This is an ID of the connection maintained by the service provider of GCC, on which the event in question is received or sent.

crntCallState

The current state of the GCC for the call.

evntPres

This flag indicates whether the detailed dump of the event, received or sent, is available. The presence of this flag indicates that **evnt** has a valid pointer.

***evnt**

Pointer to the event structure. It has the format:

```
typedef union ccAllObsTrcEvnts
{
   CcAllSdus   ccEvnt;
   CcRtEvnt    rtEvnt;
   CcRmEvnt    rmEvnt;
   CcSfEvnt    sfEvnt;
   CcTmrEvnt   tmrEvnt;
} CcAllObsTrcEvnts;
```

ccEvnt

Refer to the *CCT Interface Service Definition* for a detailed description of the following data structure.

```
typedef struct ccAllSdus           /* all sdu messages */
{
   union
   {
      CcConEvnt    ccConEvnt;      /* Connect Event */
      CcCnStEvnt   ccCnStEvnt;     /* Connect Status Event */
      CcRelEvnt    ccRelEvnt;      /* Release Event */
      CcMntStaEvnt ccMntStaEvnt;   /* Maintenance Status Event
                                   */
      CcHldEvnt    ccHldEvnt;      /* Hold Event */
      CcRtrEvnt    ccRtrEvnt;      /* Retreive Event */
      RmRsc        ccRscEvnt;
   } m;
} CcAllSdus;
```

rtEvnt

Refer to the section on the Router in the *Interworking Call Control Interface Service Definition* for a detailed description of the following data structure.

```
typedef union ccRtEvnt
{
   RtRteReqEvnt  rteReqEvnt;
   RtRteCfmEvnt  rteCfmEvnt;
   RtRelEvnt     rteRelEvnt;
   RtRspEvnt     rteRspEvnt;
   RmInterface   interface;
} CcRtEvnt;
```

rmEvnt

Refer to the section on the Router in the *Interworking Call Control Interface Service Definition* for a detailed description of the following data structure.

```
typedef union ccRmEvnt
{
   RmAlocReqEvnt     alocReqEvnt;
   RmAlocCfmEvnt     alocCfmEvnt;
   CcRmDealocReqEvnt dealocReqEvnt;
   CcRmDealocCfmEvnt dealocCfmEvnt;
   CcRmDealocIndEvnt dealocIndEvnt;
} CcRmEvnt;
```

sfEvnt

Refer to the section on the Router in the *Interworking Call Control Interface Service Definition* for a detailed description of the following data structure.

```
typedef struct ccSfEvnt
{
   RmRsc       *resource1;
   RmRsc       *resource2;
   RmTfcDesc   *tfcDsc;
   U8          swtchResult;
   SwtchIdx    swtchIdx;
```

tmrEvnt

```
typedef struct ccTmrEvnt
{
   U8 tmrType;
} CcTmrEvnt;
```

tmrType

Timer ID.

nl

This is a dummy structure included to prevent a null union definition for the **trc** structure, in case the observation trace feature is not enabled.

Description:

GCC uses this function to provide the layer manager with trace information. Trace generation can be enabled or disabled via the layer manager control request.

Returns:

00 **ROK**

01 **RFAILED**

3.5.1.2.11 CcMiLccAcntInd

Name:

Accounting Indication

Direction:

GCC to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 CcMiLccAcntInd(pst, acnt)
Pst     *pst;
CcMngmt *acnt;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

acnt

Pointer to the accounting information structure. It has the following format:

```
typedef struct ccMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
/* Accounting Indication */
      struct
      {
         DateTime       dt;                 /* date and time */
         CcCallDtlRcrd  callDtlRcrd;        /* Call Detail Record */
      } acnt;                               /* Accounting indication */
   }t;
} CcMngmt;
```

hdr

Header structure. For more details, see Section 3.3.1, "Header."

cfm

The status field is not significant in this primitive.

dt

Date and time structure.

callDtlRcrd

This structure contains the call detail record for the call.

```
typedef struct ccCallDtlRcrd
{
   U8          cdrStatus;      /* Call is released/ Call in progress */
   U8          callState;      /* State of the call before
                                * call started clearing
                                */
   RtRoute     cdPtyNmb;       /* Called Party Number */
   U8          cgPtyNmbPres;   /* Calling Party Number Present */
   RtRoute     cgPtyNmb;       /* Calling Party Number */
   U8          cllngCtgy;      /* Calling party category */
   U8          cgNmbIncpltInd; /* calling party number incomplete
                                  indicator */
   U8          redirgNumPres;  /* Calling Party Number Present */
   RtRoute     redirgNum;      /* Redirecting number */
   RmRsc       origRsc;        /* Originating Resource */
   RmRsc       termRsc;        /* Terminating Resource */
   RmTfcDesc   tfcDesc;        /* Generic Traffic Descriptor */
   DateTime    tmIcConIndRcvd; /* Time Incoming ConInd received */
   TknU32     tckIcConIndRcvd; /* Ticks Incoming ConInd received */
   TknU32     tckOgConReqSent; /* Ticks Outgoing Connection
                                * Request sent
                                */
   TknU32     tckOgConCfmRcvd; /* Ticks Outgoing Connection
                                * Confirm received
                                */
   TknU32     tckIcConRspSent; /* Ticks Incoming Connection
                                * Response sent
                                */
   TknU32     tckRelIndRcvd;   /* Ticks Release Indication
                                * received
                                */
   TknU32     tckRelReqSent;   /* Ticks Release Request sent */
   Cntr       icUUMsgPriorAnswered; /* UU messages over incoming
                                       interface
                                     * prior to call being answered
                                     */
   Cntr       icUUMsgAnswered; /* UU messages over incoming interface
                                * after call being answered
                                */
   Cntr       ogUUMsgPriorAnswered; /* UU messages over outgoing
                                       interface
                                     * prior to call being answered
                                     */
   Cntr       ogUUMsgAnswered; /* UU messages over outgoing interface
                                * after call being answered
                                */
   U8         end2endIERcvdInd; /* end-to-end information indicator */
   CcCause     relCause;       /* Release cause */
   U8          relOrign;       /* Release Origin */
} CcCallDtlRcrd;
```

cdrStatus

Indication if the call is still active or released. An indication, if enabled, is generated at the end of each call with the **cdrStatus** set to the released call. An indication for the active call is generated upon timer expiration, **tCallDtl** (refer to the general configuration in the **CcMiLccCfgReq**).

Value	Description
LCC_CDR_CALLCLEARED	The call is being cleared.
LCC_CDR_CALLINPROGRESS	The call is in progress.

callState

The call state. This indicates whether the call is cleared before or after it is answered.

Value	Description
LCC_CDR_UNANSWERED	The call is cleared before being answered.
LCC_CDR_ANSWERED	The call is cleared after being answered.

cdPtyNmb

Called party number. For more information, refer to Section 3.3.9, "Route Structure."

cgPtyNmbPres

Flag to indicate whether the calling party number is valid. The allowable values are:

PRSNT_NODEF
NOTPRSNT

Note: **PRSNT_NODEF** *means that the* **cgPtyNmb** *is valid.*

cgPtyNmb

Calling party number. For more details, refer to Section 3.3.9, "Route Structure."

cllngCtgy

Calling party category. The allowable values are:

CC_CAT_UNKNOWN
CC_CAT_OPLANGFR
CC_CAT_OPLANGENG
CC_CAT_OPLANGGER
CC_CAT_OPLANGRUS
CC_CAT_OPLANGSP
CC_CAT_ADMIN1
CC_CAT_ADMIN2
CC_CAT_ADMIN3
CC_CAT_ORD
CC_CAT_PRIOR
CC_CAT_DATA
CC_CAT_TEST
CC_CAT_PAYPHONE

cgNmbIncpltInd

Calling party number complete indicator. Currently, it is zero (0).

redirgNmbPres

Flag to indicate whether the redirection number is valid. The allowable values are:

PRSNT_NODEF
NOTPRSNT

Note: **PRSNT_NODEF** *means that the* **redirgNmb** *is valid.*

redirgNmb

Redirection number. For more information, refer to Section 3.3.9, "Route Structure."

origRsc

Originating resource. For more information, refer to Section 3.3.8, "Network Resource."

termRsc

Terminating resource. For more information, refer to Section 3.3.8, "Network Resource."

tfcDesc

Generic traffic descriptor. For more information, refer to Section 3.3.10, "Traffic Descriptor."

tmIcConIndRcvd

Time incoming **ConInd** is received. This is the absolute time when the incoming connection indication was received.

tckIcConIndRcvd

System ticks at the time when the incoming connection indication was received.

tckOgConReqSent

System ticks at the time when the outgoing connection indication was sent.

tkOgConCfmRcvd

System ticks at the time when the outgoing connection confirm was received.

tkIcConRspSent

System ticks at the time when the incoming connection response was sent.

tkRelIndRcvd

System ticks at the time when the release indication was received.

tkRelReqSent

System ticks at the time when the release request was sent.

icUUMsgPriorAnswered

Number of user-to-user messages received at the incoming side, prior to the answered state.

icUUMsgAnswered

Number of user-to-user messages received at the incoming side, after reaching the answered state.

icUUMsgPriorAnswered

Number of user-to-user messages received at the outgoing side, prior to the answered state.

icUUMsgAnswered

Number of user-to-user messages received at the outgoing side, after reaching the answered state.

end2endIERcvdInd

Information about transmitted end-to-end information. A bit is assigned to each type of end-to-end information. When one of the following end-to-end information is received, the bit is set.

CC_CDR_NBNHLINFO
CC_CDR_NBNLLINFO
CC_CDR_PROGIND
CC_CDR_CGPTYSAD
CC_CDR_CDPTYSAD
CC_CDR_NBBEARCAP
CC_CDR_MOREDATA
CC_CDR_ALLE2EINFO
CC_CDR_CNPTYSAD

relCause

Cause of the call release. For more details, refer to Section 3.3.11, "Cause."

relOrign

The side at which the call release was originated.

Value	Description
LCC_RELORGN_IN	Incoming side originated the release
LCC_RELORGN_OUT	Outgoing side originated the release
LCC_RELORGN_INTERNAL	ICC internally initiated the release

Description:

GCC uses this function to provide the layer manager with the accounting information for the call. Generating accounting information can be enabled or disabled via the layer manager control request.

Returns:

00 **ROK**

01 **RFAILED**

3.5.2 Interface with the Lower Layers

This section discusses GCC's interface with the lower layers.

3.5.2.1 General

GCC is the service user of different lower layers. Primitives are provided with different prefixes depending on the lower layer.

Name	Description
`CcLiCct`	Lower layer interface with the PSIF(s)
`CcLiSft`	Lower layer interface with the SFM
`CcLiRmt`	Lower layer interface with the RM
`CcLiRtt`	Lower layer interface with the static RT
`PuLiPci`	Lower layer interface with the PN (Trillium's PNNI router)

3.5.2.2 PSIF Interface

The following is a list of primitives used between GCC and the PSIF. The XM also uses this interface to establish and terminate calls. For a detailed description, refer to the *CCT Interface Service Definition*.

Bind Establishment

Primitive Name	Description	Flow
`XxYyCctBndReq`	Bind request	GCC to PSIF
`XxYyCctBndCfm`	Bind confirm	PSIF to GCC

Generic Call Control

Primitive Name	Description	Flow
`XxYyCctConInd`	Connection establishment indication	PSIF to GCC
`XxYyCctConReq`	Connection establishment request	GCC to PSIF
`XxYyCctConCfm`	Connection establishment confirm	PSIF to GCC
`XxYyCctConRsp`	Connection establishment response	GCC to PSIF
`XxYyCctAddrInd`	Additional addressing indication	PSIF to GCC
`XxYyCctRscCfm`	Resource confirm	PSIF to GCC
`XxYyCctRscRsp`	Resource response	GCC to PSIF
`XxYyCctCnStInd`	Connection status indication	PSIF to GCC

Primitive Name	Description	Flow
XxYyCctCnStReq	Connection status request	GCC to PSIF
XxYyCctRelInd	Release indication	PSIF to GCC
XxYyCctRelReq	Release request	GCC to PSIF
XxYyCctRelRsp	Release response	GCC to PSIF
XxYyCctRelCfm	Release confirm	PSIF to GCC
XxYyCctStaInd	Status indication	PSIF to GCC
XxYyCctModInd	Modification indication	PSIF to GCC
XxYyCctModReq	Modification request	GCC to PSIF
XxYyCctModRsp	Modification response	GCC to PSIF
XxYyCctModCfm	Modification confirm	PSIF to GCC
XxYyCctHldInd	Connection hold indication	PSIF to GCC
XxYyCctRtrInd	Connection retrieve indication	PSIF to GCC
XxYyCctProfInd	Profile indication	PSIF to GCC
XxYyCctStaReq	Status request	GCC to PSIF

Circuit Supervision

Primitive Name	Description	Flow
XxYyCctMntStaInd	Maintenance status indication	PSIF to GCC
XxYyCctMntStaReq	Maintenance status request	GCC to PSIF

3.5.2.3 Interface with the Router

The router routes to GCC. GCC supports routing, via both a static router (RT) and Trillium's PNNI router (PN).

3.5.2.3.1 Interface with the Static Router (RT)

GCC's interface with the static router is known as the RTT interface. The following primitives are provided at this interface.

Bind Establishment

Primitive Name	Description	Flow
XxYyRttBndReq	Bind request	GCC to RT
XxYyRttBndCfm	Bind confirm	RT to GCC

Route Determination

Primitive Name	Description	Flow
XxYyRttRteReq	Route request	GCC to RT
XxYyRttRteCfm	Route confirm	RT to GCC
XxYyRttRteRsp	Route response	GCC to RT
XxYyRttRelReq	Route release request	GCC to RT
XxYyRttRelInd	Route release indication	RT to GCC
XxYyRttMntStaReq	Maintenance status request	GCC to RT
XxYyRttMntStaInd	Maintenance status indication	RT to GCC

For a detailed description of the RTT interface, refer to the *Interworking Call Control Interface Service Definition*

3.5.2.3.2 Interface with Trillium's PNNI Router (PN)

The PCI interface is the interface between GCC and Trillium's PNNI Router. The following primitives are provided at this interface.

Bind Establishment

Primitive Name	Description	Flow
XxYyPciBndReq	Bind request	GCC to PN

Route Determination

Primitive Name	Description	Flow
`XxYyPciRteReq`	Route request	GCC to PN
`XxYyPciRteCfm`	Route confirm	PN to GCC
`XxYyPciRteRsp`	Route response	GCC to PN
`XxYyPciRelReq`	Route release request	GCC to PN
`XxYyPciRelInd`	Route release indication	PN to GCC

For a detailed description of the PCI interface, refer to the *PNNI Service Definition*.

3.5.2.4 Interface with the Resource Manager and Connection Manager

The following primitives are provided at GCC's interface with the RM, which is called the RMT interface.

Bind Establishment

Primitive Name	Description	Flow
`XxYyRmtBndReq`	Bind request	GCC to RM
`XxYyRmtBndCfm`	Bind confirm	RM to GCC

Resource Management

Primitive Name	Description	Flow
`XxYyRmtAlocReq`	Resource allocation request	GCC to RM
`XxYyRmtAlocCfm`	Resource allocation confirm	RM to GCC
`XxYyRmtModReq`	Resource modification request	GCC to RM
`XxYyRmtModCfm`	Resource modification confirm	RM to GCC
`XxYyRmtDalocReq`	Resource deallocation request	GCC to RM
`XxYyRmtDalocCfm`	Resource deallocation confirm	RM to GCC
`XxYyRmtGrpAlocReq`	Resource group allocation request	PSIF to RM
`XxYyRmtGrpDalocReq`	Resource group deallocation request	PSIF to RM
`XxYyRmtDealoInd`	Resource deallocation indication	RM to GCC
`XxYyRmtAudReq`	Audit request	RM to GCC
`XxYyRmtAudCfm`	Audit confirmation	GCC to RM

Auditing

Primitive Name	Description	Flow
`XxYySftAudReq`	Switching audit request	GCC to SFM
`XxYySftAudCfm`	Switching audit confirm	SFM to GCC

For a detailed description of the RMT interface, refer to the *Interworking Call Control Interface Service Definition*.

3.5.2.5 Interface with the Switching Fabric Manager

The following primitives are provided at GCC's interface with the SFM, which is called the SFT interface.

Bind Establishment

Primitive Name	Description	Flow
`XxYySftBndReq`	Bind request	GCC to SFM
`XxYySftBndCfm`	Bind confirm	SFM to GCC

Switching Establishment and Release

Primitive Name	Description	Flow
`XxYySftConReq`	Switching connect request	GCC to SFM
`XxYySftConCfm`	Switching connect confirm	SFM to GCC
`XxYySftRelReq`	Switching release request	GCC to SFM
`XxYySftRelCfm`	Switching release confirm	SFM to GCC
`XxYySftRelInd`	Switching release indication	SFM to GCC
`XxYySftAudReq`	Switching audit request	GCC to SFM
`XxYySftAudCfm`	Switching audit confirm	SFM to GCC

Auditing

Primitive Name	Description	Flow
`XxYySftAudReq`	Switching audit request	GCC to SFM
`XxYySftAudCfm`	Switching audit confirm	SFM to GCC

For a detailed description of the SFT interface, refer to the *Interworking Call Control Interface Service Definition*.

3.5.3 Interface with System Services

This section discusses GCC's interface with system services.

3.5.3.1 General

This section describes the system services required by GCC.

Task Scheduling

The task scheduling management functions are called to register, activate, and terminate a task. Use the following functions for task scheduling management.

Name	Description
`SRegActvTsk`	Registers activate task
`ccActvTsk`	Activates task for GCC
`SPstTsk`	Post task
`SExitTsk`	Exit task

Initialization

The Operating System (OS) calls the initialization management function to initialize a task. Use the following function for initialization management.

Name	Description
`ccActvInit`	Activates task - Initialize GCC

Memory Management

The memory management functions allocate and deallocate variable-sized buffers that are static buffers. Use the following functions for memory management.

Name	Description
`SGetSBuf`	Get static buffer
`SGetSMem`	Get static memory

Message Management

The message management functions initialize, add data to, and remove data from messages by utilizing dynamic buffers. Use the following functions for message management.

Name	Description
`SGetMsg`	Allocates a message (from a dynamic pool)
`SPutMsg`	Deallocates a message (into a dynamic pool)
`SFndLenMsg`	Finds the length of a message
`SExamMsg`	Examines an octet at a specified index in a message
`SRepMsg`	Replaces an octet at a specified index in a message
`SAddPstMsg`	Adds an octet to the end of a message
`SRemPreMsg`	Removes an octet from the beginning of a message
`SAddPreMsgMul`	Adds multiple octets to the beginning of a message
`SRemPreMsgMult`	Removes multiple octets from the beginning of a message
`SRemPstMsgMult`	Removes multiple octets from the end of a message
`SPkS8`	Adds a signed 8-bit value to a message
`SPkU8`	Adds an unsigned 8-bit value to a message
`SPkS16`	Adds a signed 16-bit value to a message
`SPkU16`	Adds an unsigned 16-bit value to a message
`SPkS32`	Adds a signed 32-bit value to a message
`SPkU32`	Adds an unsigned 32-bit value to a message
`SUnpkS8`	Removes a signed 8-bit value from a message
`SUnpkU8`	Removes an unsigned 8-bit value from a message
`SUnpkS16`	Removes a signed 16-bit value from a message
`SUnpkU16`	Removes an unsigned 16-bit value from a message
`SUnpkS32`	Removes a signed 32-bit value from a message
`SUnpkU32`	Removes an unsigned 32-bit value from a message

Timer Functions

Name	Description
`SRegTmr`	Registers activation function - Timer
`SDeRegTmr`	Deregisters activation function - Timer
`ccPrcConTq`	Timer activation function for GCC. This is used for connection-related timers (inter-digit timer, setup timer).
`ccPrcSapTq`	Timer activation function for GCC. This is used for SAP-related timers (bind confirm timer).

Miscellaneous

Resource availability checking. The following miscellaneous functions are used.

Name	Description
`SFndProcId`	Finds the processor ID on which a task runs
`SGetDateTime`	Gets real date and time
`SLogError`	Handles an error
`SPrint`	Prints a preformatted string to the default display device

For a detailed description of these system services, refer to the *System Services Interface Service Definition*.

3.5.3.2 ccActvInit

Name:

Activate Task - Initialize GCC

Direction:

System services to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ccActvInit(ent, inst, region, reason)
Ent    ent;
Inst   inst;
Region region;
Reason reason;
```

Parameters:

ent

Entity ID.

inst

Instance ID for the entity.

region

Memory region ID used by the layer to get static memory.

reason

Reason for initialization. This field is not currently used.

Description:

System services uses this function to initialize GCC.

Returns:

00 **ROK**

01 **RFAILED**

3.5.3.3 ccActvTsk

Name:

Activate Task

Direction:

System services to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ccActvTsk(pst, mBuf)
Pst    *pst;
Buffer *mBuf;
```

Parameters:

`pst`

For more description, see Section 3.3.3, "**`Pst`**."

`mBuf`

Message buffer.

Description:

System services calls this function, which injects an event and a primitive into the GCC layer. The given message buffer is unpacked to find the corresponding primitive and associated parameters. Then, the appropriate primitive reception handler is scheduled.

Returns:

`00     ROK`

`01     RFAILED`

3.5.3.4 ccPrcConTq

Name:

Activate Task - Timer

Direction:

System services to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ccPrcConTq()
```

Parameters:

None

Description:

System services uses this function to activate GCC timers with a timer tick. While it processes the general configuration request, the GCC protocol layer registers this function with system services. The protocol layer uses the **SRegTmr** system services primitive and passes the pointer to **ccPrcConTq** as an argument to register the GCC timer function with system services. The period during which this timer function must be invoked is also passed in the **SRegTmr**. The **ccPrcConTq** function processes timers on a per-connection basis (setup timer, release timer).

Returns:

00 **ROK**

01 **RFAILED**

3.5.3.5 ccPrcSapTq

Name:

Activate Task - Timer

Direction:

System services to GCC

Supplied:

Yes

Synopsis:

```
PUBLIC S16 ccPrcSapTq()
```

Parameters:

None

Description:

System services uses this function to activate GCC timers with a timer tick. While it processes the general configuration request, the GCC protocol layer registers this timer function with system services. The protocol layer uses the **SRegTmr** system services primitive and passes the pointer to the **ccPrcSapTq** as an argument to register the GCC timer function with system services. The period during which this timer function must be invoked is also passed in the **SRegTmr**. The **ccPrcSapTq** function processes timers on a per-SAP basis (wait for the bind confirm).

Returns:

00	**ROK**
01	**RFAILED**

3.6 Static Router

This section discusses the static router and discusses its interfaces and associated primitives.

3.6.1 Interface with the Layer Manager

This section discusses RT's interface with its layer manager (LRT).

3.6.1.1 Primitive Overview

The following is a list of primitives used between RT and its layer manager.

Configuration

The following functions configure protocol layer resources.

Name	Description
`RtMiLrtCfgReq`	Configuration request
`RtMiLrtCfgCfm`	Configuration confirm

Control

The following primitives control the RT.

Name	Description
`RtMiLrtCntrlReq`	Control request
`RtMiLrtCntrlCfm`	Control confirm

Statistics

The following primitives retrieve statistics information.

Name	Description
`RtMiLrtStsReq`	Statistics request
`RtMiLrtStsCfm`	Statistics confirm

Solicited Status

The following primitives retrieve the status of internal RT information.

Name	Description
RtMiLrtStaReq	Status request
RtMiLrtStaCfm	Status confirm

Unsolicited Status

The RT uses the following function to indicate status changes.

Name	Description
RtMiLrtStaInd	Status indication

3.6.1.2 Specific

This section discusses in detail the specific primitives exchanged by the RT and layer manager.

3.6.1.2.1 RtMiLrtCfgReq

Name:

Configuration Request

Direction:

Layer manager to the RT

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RtMiLrtCfgReq(pst, cfg)
Pst     *pst;
RtMngmt *cfg;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. The configuration structure has the following format:

```
typedef struct rtMngmt
{
   Header    hdr;                   /* header */
   CmStatus  cfm;                   /* status in confirm */
   union
   {
/* configuration */

      struct
      {
         union
         {

            RtGenCfg  rtGen;      /* General Config */
            RtSapCfg  rtSap;      /* SAP Config */
            RtRoutCfg rtRout;     /* Route Config */
            RtIntfCfg rtIntf;     /* Interface Config */
            RtVirIntfcCfg rtVirIntfc; /* Virtual interface config */
          } s;
      } cfg;                        /* configuration */
   } t;
} RtMngmt;
```

hdr

For more description, see Section 3.3.1, "Header."

cfm

Status field. Only the confirmation primitives use this field to report errors. It is not significant to the configuration request.

rtGen

General RT configuration structure. The general configuration must be done first. The RT uses much of the information carried by this table to reserve a proper amount of necessary static memory.

```
typedef struct rtGenCfg         /* general configuration */
{
   U16   nmbSaps;               /* Number of Saps */
   U16   nmbRoutes;             /* Number of routes */
   U16   nmbRoutesAlternate;    /* maximum number of routes which
                                   can have route + interface based
                                   digit stripping */
   U32   nmbIntfc;              /* Number of interfaces */
   U32   nmbTrnkdIntfc;         /* Number of trunked or
                                 * virtual interfaces
                                 */
   U32   trnkdIntfcHlSz;        /* Maximum number of bins in the
                                 * trunked interface hash list
                                 */
   U8    maxRouteLen;           /* Route length */
   Pst   sm;                    /* Stack manager */
#ifdef ICC_AUDIT
   S16   timeRes;               /* timer Resolution */
#endif /* ICC_AUDIT */
} RtGenCfg;
```

nmbSaps

Number of upper SAPs. The upper SAP is the SAP toward GCC. The number of upper SAPs corresponds to the number of GCCs in the system. The allowable values are: 1 to 32767.

nmbRoutes

Maximum number of routes configured in the system.
The allowable values are: 1 to 32767.

nmbRoutesAlternate

With each route configured in the RT, you can configure which type of digit stripping method is required. The choice of a particular digit stripping method affects the amount of memory required in the system. If the routes use the route and the interface **LRT_METHOD_ROUTNINTFC-**based digit stripping, specify the number of such possible routes here. The allowable values are: 1 to 32767.

nmbIntfc

Maximum number of interfaces configured in the RT.
The allowable values are: 1 to **MAXNUMINTF**; and **MAXNUMINTF** can be 1 to 65535.

nmbTrnkdIntfc

Number of trunked or virtual interfaces configured in the RT.

trnkdIntfcHlSz

Size of the trunked interface hash list. The ideal value equals the number of trunked interfaces existing in the system. In this case, each hash list bin has a maximum of one entry, and the search time is minimal. Reducing the size of the hash list increases search time, but less memory is required. There is always a trade-off between time and memory. A good value is about one-fourth the number of connections, so that a hash list bin has a maximum of four entries.

maxRouteLen

Maximum length of the route. The layer manager must configure this as the length of the longest configured route entry.

sm

Stack manager post structure. The RT requires the post structure when it sends an unsolicited status to the stack manager. An unsolicited status is sent to the address in the **sm** field.

timeRes

Timer resolution, that is, the period after which the common timer function is periodically called for this module. The module uses this period internally to maintain different timers for different connections.

rtSap

Upper SAP configuration structure. This SAP is used to communicate with GCC.

```
typedef struct rtSapCfg          /* SAP config */
{
   SpId      spId;               /* service provider id, SAP id */
   Priority  prior;              /* priority */
   Route     route;              /* route */
   Selector  selector;           /* selector */
   MemoryId  mem;                /* memory region & pool id */
#ifdef ICC_AUDIT
   U8 cid;                       /* call control Id */
   RtSapTmrCfg tmr;              /* SAP timers */
#endif /* ICC_AUDIT */
} RtSapCfg;
```

spId

Service provider ID. The RT uses this **spId** to identify the SAP on which it communicates with GCC.

prior

Priority used when the task buffers must be posted between the service provider and service user. It is used only in a loosely coupled system.
The allowable value: **PRIOR0**.

route

The system uses this for internal routing requirements. TAPA does not define the contents or the use of this information. It is used only in a loosely coupled system.

The allowable value: **RTESPEC—route to specific instance**.

selector

Defines whether the service provider and service user are loosely or tightly coupled. It is used to resolve a primitive call to the upper layer. The allowable values depend on the configuration. For more details, refer to the *Interworking Call Control Portation Guide*.

mem

For further description, refer to Section 3.3.5, "Memory."

cid

ID of the GCC entity, which binds to this SAP.

tmr

SAP **timres**, which is defined as:

```
typedef struct RtSapTmrCfg
{
   TmrCfg audCfmTmr;       /* timer for waiting for Audit Confirm
                                 for PAP */
   TmrCfg periodAudTmr;    /* timer Periodical Auditing*/
} RtSapTmrCfg;
```

audCfmTmr:

Timer to wait for the audit confirmation. For further details, refer to Section 3.3.4, "Timer Configuration."

periodAudTmr

Periodic auditing timer. For details, refer to Section 3.3.4, "Timer Configuration."

rtRout

Route configuration structure.

```
typedef struct rtRoutCfg        /* Route Configuration */
{
   RtRoute     rt;              /* Routing info */
   U8          vpnId;           /* Identifier for the VPN */
   U8          routLoc;         /* location of the route
                                 * address
                                 */
   S16         nmbIntfc;        /* number of IF leading to a specific
                                   rout */
   RmInterface intfc[LRT_MAXNUMINTF]; /* destination point code */
   U8          strpType;        /* type of digit stripping */
   union
   {
      RtDgtsStrpInfo  rtDgtsStrpInfoS; /* simple method */
      RtDgtsStrpInfo  rtDgtsStrpInfoC[LRT_MAXNUMINTF]; /* complete
                                                          method */
   }rtDgtsStrp;
} RtRoutCfg;
```

rt

Identifies the route configured.

```
typedef struct rtRoute          /* Route information */
{
   U8          addrInd;         /* type of address */
   U8          nmbDigits;       /* number of digits */
   U8          numPlan;         /* numbering plan */
   U8          ident;           /* identification */
   Addrs       addr;            /* routing part of the address */
} RtRoute;
```

addrInd

Identifies the address type, for example, if the called party number or the transit network selection has been specified as the route. The allowable values are:

Value	Description
CC_CDPTY	Called party number
CC_TRANNET	Transit network selection

nmbDigits

Number of digits in the route. It identifies the number of digits in the **addr** field. Because the digits are stored in the **addr** in BCD form, as described in the ISUP/B-ISUP protocol, the **nmbDigits** contains the number of digits and not the number of valid octets compared in the **addr** field.

numPlan

If the **addrInd** indicates the called party number, this field contains the numbering plan identification. In case of transit network selection, this field contains the network identification plan. The allowable values are:

Value	Description
CC_NP_UNK	Number not present—the **np** is set to zero.
CC_ISDNNUM	ISDN numbering plan (CCITT E.164)
CC_DATANUM	Data numbering - X.121
CC_TELEXNUM	Telex numbering - Recommendation F.69
CC_PRIVATENUMPLAN	Private numbering pan
CC_UNKNOWNPLAN	Unknown plan
CC_TELEPNUMPLAN	Telephony numbering plan (CCITT E.163) identification

If the **addrInd** indicates that the route is the called party number type, this field corresponds to the nature of the address indicator.

CC_NA_SUBSNUM
CC_NA_UNKNOWN
CC_NA_NATNUM
CC_NA_INTNATNUM
CC_NA_NSPNUM

If the **addrInd** indicates that the route corresponds to the transit network selection, this field contains the type of network identification field.

addr

The string of digits corresponding to the route or to the transit network selection digits. The digits are stored as BCD digits, with two BCD digits packed in one octet.

vpnId

Each VPN is assigned a unique ID. The router requires the **vpnId** to identify the route in case of private numbering plans. **vpnId** 0 is reserved for the public network. The allowable values are: 0 to 99.

routLoc

Location of the routing address. The allowable values are:

RT_OUTGOING
RT_SUBSCRIBER
RT_INTERNAL

When the value is **RT_INTERNAL**, the route must examine the calling party number to find exactly to which local interface, out of the many specified in the route entry, applies to this call.

nmbIntfc

Number of interfaces associated with this route.
The allowable values are: 0 to **nmbIntfc** in the **rtGenCfg**.

intfc

An array of interfaces associated with a route. For further information, see Section 3.3.6, "Interface."

strpType/rtDgtsStrp

The **strpType** field indicates the type of digit stripping from the called party number employed for this route. The corresponding member in the **rtDgtsStrp** should be filled depending on this value.

```
typedef struct rtDgtsStrpInfo    /* Digit Stripping information */
{
   U8          numDgts;          /* number of digits */
   U8          ident;            /* identification */
} RtDgtsStrpInfo;
```

numDgts

This indicates the number of digits to be stripped from the called party number.

ident

This field indicates the modified type of number. This field takes the same values, as described above in this section.

The **strpType** field can take following values:

LRT_NO_METHOD
LRT_METHOD_ROUTONLY
LRT_METHOD_ROUTNINTFC

If the **strpType** value is **LRT_NO_METHOD**, digit stripping is not required, and thus, the called party number is passed as received to the next node.

If the **strpType** value is **LRT_METHOD_ROUTONLY**, it indicates that digit stripping should be done based on this route only, irrespective of the interface selected. The **rtDgtsStrpInfoS** member of the union should be filled with proper values.

If the **strpType** value is **LRT_METHOD_ROUTNINTFC**, it indicates that digit stripping should be done based on the route and interface selected. The **rtDgtsStrpInfoC** member of the union should be filled with proper values.

rtIntf

Each interface used in the route configuration as indicated below must be configured prior to the route configuration. When a PNNI router is used for routing with the static router, a PNNI interface with a dummy interface ID (**RT_DUMMY_INTFCID**) must be configured in the static router. The static router selects and returns this interface in the **RteCfm** primitive for all calls to be rerouted using the PNNI router. For an illustration, refer to Figure 4-38.

```
RT_DUMMY_INTFCID  0xFFFFFFFF

typedef struct rtIntfcCfg       /* Interface configuration */
{
   U8          vpnId;           /* Identifier for the VPN */
   RmInterface intfc;           /* Interface being configured */
   U8          minDgtsToSeize;  /* Minimum Digits to Seize */
   U8          protType;        /* Protocol and variant used */
   U8           call1;          /* number of calls (in 10%)
                                   accepted on */
   U8          call2;           /* the appropriate congestion level */
   U8          call3;
   U8          prior1;          /* If priority calls can be routed on */
   U8          prior2;          /* the appropriate congestion level */
   U8          prior3;
   U8          areaCode[MAXACODESIZE]; /* area code */
} RtIntfCfg;
```

vpnId

ID of the VPN.

intfc

This identifies the configured interface.

minDgtsToSeize

Minimum number of digits that must be present in the called party number before issuing a connect request.

protType

Identifies the protocol and its variant used at this interface. For a list of allowable values, refer to Section 3.3.7, "Protocol Variants."

call1, call2, call3

Percentage of non-priority calls allowed during the specified congestion level. Currently, three levels of congestion are supported. Value 1 for parameter 1 means that ten percent of the calls are allowed. Ten would mean that 100 percent of the calls are allowed. The allowable values are: 0 to 9.

prior1, prior2, prior3

Indicates whether priority calls are allowed during the specified congestion level. Currently, three levels of congestion are supported. These fields have boolean values indicating whether all calls or none of the priorities are allowed during congestion. The allowable values are: **TRUE**, **FALSE**.

areaCode

This field provides the area code associated with the incoming call at this interface. This field is used to prefix the area code to the received calling party number. The **areaCode** is supplied as a NULL-terminated ASCII string.

The allowable value: **MAXACODESIZE 8.**

rtVirIntf

The virtual interface configuration is required to be configured for the feature transparency or trunking solutions—an example is ISDN over Q.SAAL. These interfaces are trunked over another interface. The information corresponding to the interface that provides trunking is prefixed with **trnkg**, while configuring the virtual interface.

```
typedef struct rtVirIntfcCfg     /* Interface configuration */
{
   U8          vpnId;            /* Identifier for the VPN */
   RmInterface intfc;            /* Interface being configured */
   U8          minDgtsToSeize;   /* Minimum Digits to Seize */
   U8          protType;         /* Protocol and variant used */
   U8          call1;            /* number of calls (in 10%)
                                    accepted on */
   U8          call2;            /* the appropriate congestion
                                    level */
   U8          call3;
   U8          prior1;           /* If priority calls can be routed
                                    on */
   U8          prior2;           /* the appropriate congestion
                                    level */
   U8          prior3;
   U8          trnkgVpnId;       /* Identifier for the VPN */
   U8          trnkgMinDgtsToSeize;/* Minimum Digits to Seize */
   U8          trnkgProtType;    /* Protocol and variant used */
   U8          trnkgIntfType;    /* Identifies the trunking
                                    Interface type */
   RtRoute     lclPtyNum;        /* Called party number associated
                                    with the interface */
   RtRoute     remPtyNum;        /* Calling party number associated
                                    with the interface */
} RtVirIntfcCfg;
```

vpnId

ID of the VPN.

intfc

This identifies the configured interface.

minDgtsToSeize

Minimum number of digits that must be present in the called party number before issuing a connect request.

protType

Identifies the protocol and its variant used at this interface. For a list of allowable values, refer to Section 3.3.7, "Protocol Variants."

call1, call2, call3

Percentage of non-priority calls allowed during the specified congestion level. Currently, three levels of congestion are supported. Value 1 for parameter 1 means that ten percent of the calls are allowed. Ten means that 100 percent of the calls are allowed. The allowable values are: 0 to 9.

prior1, prior2, prior3

Indicates whether priority calls are allowed during the specified congestion level. Currently, three levels of congestion are supported. These fields have boolean values indicating whether all calls or none of the priorities are allowed during congestion. The allowable values are: **TRUE**, **FALSE.**

trknkgVpnId

This is a **vpnId** of the interface that provides trunking. This may be a public or private network. The allowable values are: 0 to 255.

trknkgMinDgtsToSeize

This is a **MinDgtsToSeize** of the interface that provides trunking.

trnkgProtType

This is a **ProtType** of the interface, which provides trunking.

It identifies the protocol and its variant used at this interface. The allowable values are:

CC_FEATTRP_SI
CC_FEATTRP_IN
CC_PH1TK_AAL1
CC_PH1TK_AAL2

For more details, refer to Section 3.3.7, "Protocol Variants."

trknkgIntfType

This is the **IntfType** of the interface providing trunking. The allowable values are:

CC_FEATTRP_INTFC
CC_PH1TK_INTFC

lclPtyNum

Local address. For more information, refer to **typedef struct rtRoute /* Route information */** on page 132.

remPtyNum

Remote address. Refer to **typedef struct rtRoute /* Route information */** on page 132 for more information.

Description:

The layer manager uses this function to configure the RT. The general configuration must be done first. Interfaces must be configured before the routes are configured. A route configuration cannot be changed. To change a route (add or remove an interface), the route must be deleted using the control request, then reconfigured.

Returns:

00 ROK

01 RFAILED

3.6.1.2.2 RtMiLrtCfgCfm

Name:

Configuration Confirm

Direction:

RT to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RtMiLrtCfgCfm(pst, cfg)
Pst     *pst;
RtMngmt *cfg;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. With the exception of the following fields, the structure used for the configuration confirm is the same as that for the configuration request. For more information, refer to Section 3.6.1.2.1, "**RtMiLrtCfgReq**."

cfm

Status field. It indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
    U16    status;              /* Status of the operation  */
    U16    reason;              /* If failed, the reason    */
} CmStatus;
```

status

It is used to return the status of the previous configuration request primitive and contains one of the following values.

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_INVALID_ELMNT	Invalid element specified in the configuration request
LCM_REASON_RECONFIG_FAIL	Reconfiguration failed
LCM_REASON_MEM_NOAVAIL	Memory allocation failed
LRT_REASON_HASHINIT_FAILED	Initialization of the hash list failed
LCM_REASON_GENCFG_NOT_DONE	Configuration request received without previous general configuration
LCM_REASON_INVALID_SAP	Invalid SAP value passed. The passed SAP does not exist in the system.
LRT_REASON_INVALID_INTERFACE	Number of interfaces exceeded. The layer manager tried to configure more interfaces than specified in the general configuration (**NmbIntfc**).
LRT_REASON_INVALID_ROUTE	Number of routes exceeded. The layer manager tried to configure more routes than specified in the general configuration (**nmbRoutes**).
LRT_REASON_ROUTE_EXISTS	The route that was configured exists already.
LRT_REASON_INTERFACE_MISSING	Route refers to an interface that has not been configured. All interfaces must be configured before the route.

Note: *The configuration confirm returns the same values as those passed in the configuration request.*

Description:

The RT uses this primitive to indicate the result of a configuration request to the layer manager.

Returns:

00 ROK

01 RFAILED

3.6.1.2.3 RtMiLrtCntrlReq

Name:

Control Request

Direction:

Layer manager to the RT

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RtMiLrtCntrlReq(pst, cntrl)
Pst     *pst;
RtMngmt *cntrl;
```

Parameters:

pst

For more information, refer to Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. The control structure has the following format.

```
typedef struct rtMngmt
{
   Header    hdr;                    /* header */
   CmStatus  cfm;                    /* status in confirm */
   union
   {
      /* control */
      struct
      {
         DateTime dt;                /* date and time */
         U8 action;                  /* action */
         U8 subAction;               /* sub action */
         union
         {
            RmInterface intfc;       /* Interface */
            RtCongCntrl  congCntrl;  /* congestion control */
            RtRoutCntrl  rout;       /* Route info */
            RtObsTrc     rtObsTrc;   /* observation index */
            U32          dbgMask;    /* debug mask */
#ifdef ICC_AUDIT
            RtAuditCntrl rtAuditCntrl; /* Audit control */
#endif /* ICC_AUDIT */
            RtGrpSapCntrl rtGrpSapCntrl; /* group Sap control */
            RtSapCntrl rtSapCntrl;       /* Sap control */
         } c;
      } cntrl;                       /* control */
   }t;
} RtMngmt;
```

hdr

For more details, see Section 3.3.1, "Header." The **elmnt** field in the element ID (**elmId**) structure defines the element. For this primitive, the allowable values are:

Value	Description
STGEN	General control
LRT_ROUTE	Route control
LRT_INTF	Interface control
LRT_OBS	Observation trigger
LRT_CONG	Congestion control
LRT_VINTF	Virtual interface control

cfm

Status field. Only the confirmation primitives use this field to report errors. It has no significance to the control request.

dt

Date and time structure.

action

Specific action code. The allowable values are:

Name	Description
AENA	Enable
ADISIMM	Disable immediately
ADEL	Delete
STRTASET	Set parameters on a configured element
AGO_ACT	Go active

subAction

The allowable values are:

Name	Description
SAUSTA	Unsolicited status generation
SAELMNT	Specific element
SADBG	Debug option
SAAUD	Audit request
SAGR_DSTPROCID	Specified group elements

intfc

Interface information. An interface can be deleted only if a router does not use it. All routes using the particular interface must be deleted prior to deleting the interface. For more information, see Section 3.6.1.2.1, "**RtMiLrtCfgReq**."

congCntrl

To set the congestion level at the interface.

```
typedef struct rtCongCntrl     /* Interface configuration */
{
   RmInterface intfc;          /* Interface being configured */
   U8          call1;          /* number of calls (in 10%)
                                * accepted on the appropriate
                                * congestion level
                                */
   U8          call2;
   U8          call3;
   U8          prior1;         /* If priority calls can
                                * be routed on the appropriate
                                * congestion level
                                */
   U8          prior2;
   U8          prior3;
} RtCongCntrl;
```

intfc, call1, call2, call3, prior1, prior2, and **prior3**

Refer to the **rtIntf** on page 135 for more information.

rout

Route information.

```
typedef struct rtRoutCntrl          /* Route Configuration */
{
   RtRoute  rt;                     /* Routing info */
   U8       vpnId;                  /* Identifies the VPN */
} RtRoutCntrl;
```

rt

See **rtRout** on page 132 for more information.

vpnId

See **vpnId** on page 135 for more information.

rtObsTrc

This configuration is required for setting a trigger based on the particular route information.

```
typedef struct rtObsTrc
{
   RtRoute     rout;           /* calling party info */
   U8          vpnId;          /* Identifies the VPN */
   U8          obsIdx;         /* observation index */
} RtObsTrc;
```

rout

The called party information on which the trigger is set.

See **rtRout** on page 132 for more details.

vpnId

See **vpnId** on page 135 for more details.

obsIdx

This index determines the column in the GCC trigger table associated with this particular route entry. If an observation trigger is not required for this route, the value of **obsIdx** should be 0.

dbgMask

Bit mask of different debug classes that can be enabled or disabled. This specifies the classes of debug messages that must be controlled (enabled or disabled). The following debug classes are defined.

Name	Description
DBGMASK_UI	Upper interface debug information
DBGMASK_MI	Layer manager debug information

rtAuditCntrl

Control request for auditing.

```
typedef struct rtAuditCntrl
{
   RmInterface intfc;  /* interface */
   SpId sapId;         /* sapId */
} RtAuditCntrl;
```

intfc

Interface for which the OAP is requested. It is used only for the OAP. The PAP is always used for all the interfaces on a particular SAP, thus, the **intfc** information is never required for the PAP. For further information, see Section 3.3.6, "Interface."

sapId

The ID of the SAP for which auditing is requested. This applies to the PAP and OAP.

rtGrpSapCntrl

Group SAP control request.

```
typedef struct rtGrpSapCntrl
{
   ProcId dstProcId;
} RtGrpSapCntrl;
```

dstProcId

The destination process ID of the entity with which the group of SAPs are bound.

rtSapCntrl

SAP control request.

```
typedef struct rtSapCntrl
{
   SpId sapId;
} RtSapCntrl;
```

sapId

SAP ID of the SAP on which this control request applies.

Description:

This primitive controls the RT. The different possible operations, with the required parameters, are:

Description	subAction	action	elmnt	dbgMask	rt	intfc	rtObsTrc	congCntrl	rtSAPCntrl	rtAudCntrl	rtGrpSAPCntrl
Enable alarms	**SAUSTA**	**AENA**									
Disable alarms		**ADISIMM**									
Enable a debug class	**SADBG**	**AENA**		Debug class to enable							
Disable a debug class		**ADISIMM**		Debug class to disable							
Delete a route	**SAELMNT**	**ADEL**	**LRT_ROUTE**		X						
Delete an interface		**ADEL**	**LRT_INTF**			X					
Enable an interface		**AENA**				X					
Disable an interface		**ADISIMM**				X					
Delete a virtual interface		**ADEL**	**LRT_VINTF**			X					

Description	subAction	action	elmnt	dbgMask	rt	intfc	rtObsTrc	congCntrl	rtSAPCntrl	rtAudCntrl	rtGrpSAPCntrl
Enable a virtual interface		AENA				X					
Disable a virtual interface		ADISIMM				X					
Enable an observation index for tracing		AENA	LRT_OBS				X				
Disable an observation index for tracing		ADISIMM	LRT_OBS				X				
Modify the congestion level		STRTASET	LRT_CONG					X			
Unbind disable an SAP		AUNBND_DIS	LRT_SAP						X		
Delete an SAP		ADEL	LRT_SAP						X		
Shut down RT entity		ASHUTDOWN	STGEN								
Enable PAP Audit	SAAUD	AENA	STRTPAPAUD							X	

Description	subAction	action	elmnt	dbgMask	rt	intfc	rtObsTrc	congCntrl	rtSAPCntrl	rtAudCntrl	rtGrpSAPCntrl
Disable PAP Audit	SAAUD	ADISIMM	STRTPAPAUD							X	
Enable OAP Audit	SAAUD	AENA	STRTOAPAUD							X	
Disable OAP Audit	SAAUD	ADISIMM	STRTOAPAUD							X	
Unbind disable a group SAP	SAGR_DSTPROCID	AUBND_DIS	STGRRTSAP								X
GO Active		AGO_ACT	STGEN								

Note: *X means that this information must be specified.*

Returns:

00 **ROK**

01 **RFAILED**

3.6.1.2.4 RtMiLrtCntrlCfm

Name:

Control Confirm

Direction:

RT to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RtMiLrtCntrlCfm(pst, cntrl)
Pst     *pst;
RtMngmt *cntrl;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

cntrl

Except for the following fields, the structure used for the control confirm is the same as that for the control request. For more information, see Section 3.6.1.2.3, "**RtMiLrtCntrlReq**."

cfm

Status field. It is used to report errors. For further details, see Section 3.3.2, "Status."

status

This field indicates the status of the previous control request primitive. It contains one of the following values.

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_INVALID_ACTION	Invalid action passed in the control structure
LCM_REASON_INVALID_SUBACTION	Invalid subaction passed in the control structure
LRT_REASON_RTDELETE_FAILED	The RT is unable to delete the route from the tree. It is an internal error.
LRT_REASON_INTERFACE_USED	Interface cannot be deleted. More than one route uses this.
LRT_REASON_INVALID_INTERFACE	This is the specified interface that is not configured.
LCM_REASON_INVALID_ELMNT	Element invalid. It must be either **LRT_ROUTE** or **LRT_INTF.**
LCM_REASON_LRT_AUD_REPEAT_REQ	There are unfinished OAP or PAP in RT

Note: *The configuration confirm returns the same values as those passed in the control request.*

Description:

The RT uses this primitive to indicate the result of a control request to the layer manager.

Returns:

00 **ROK**

01 **RFAILED**

3.6.1.2.5 RtMiLrtStsReq

Name:

Statistics Request

Direction:

Layer manager to the RT

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RtMiLrtStsReq(pst, action, sts)
Pst     *pst;
Action  action;
RtMngmt *sts;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

action

Action indicator. The allowable values are:

Name	Description
0	Zero statistics counters (**ZEROSTS**)
1	Do not set the statistics counters to zero (**NOZEROSTS**)

sts

Pointer to the statistics structure. The statistics structure has the following format:

```
typedef struct rtMngmt
{
   Header hdr;
   CmStatus  cfm;                /* status in confirm */
   union
   {
/* statistics */
      struct
      {
         DateTime dt;            /* date and time */
         Duration dura;          /* duration */
         RmInterface intfc;      /* Interface */
         RtIntfcSts s;           /* statistic counters */
      }sts;
   }t;
} RtMngmt;
```

hdr

For more details, see Section 3.3.1, "Header."

cfm

Status field. Only the confirmation primitives use this field to report errors. It is not significant to the statistics request.

dt

Date and time structure.

dura

Duration structure.

intfc

Interface for which statistics are requested. For details on how to specify the interface, see Section 3.6.1.2.1, "**RtMiLrtCfgReq**."

s

Statistic counters. This field is not relevant to the statistics request.

Description:

The layer manager uses this function to gather the statistics information at a particular interface.

Returns:

00 ROK

01 RFAILED

3.6.1.2.6 RtMiLrtStsCfm

Name:

Statistics Confirm

Direction:

RT to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RtMiLrtStsCfm(pst, sts)
Pst     *pst;
RtMngmt *sts;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sts

Pointer to the statistics structure. The statistics structure has the following format.

```
typedef struct rtMngmt
{
   Header hdr;
   CmStatus  cfm;                /* status in confirm */
   union
   {
/* statistics */
      struct
      {
         DateTime    dt;          /* date and time */
         Duration    dura;        /* duration */
         RmInterface intfc;       /* Interface */
         RtIntfcSts  s;           /* statistic counters */
      }sts;
    } t;
} RtMngmt;
```

hdr

Header structure. For more description, see Section 3.3.1, "Header."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
   U16   status;              /* Status of the operation  */
   U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. For more information, see Section 3.6.1.2.3, "**RtMiLrtCntrlReq**." It contains one of the following values.

Name	Description
LCM_PRIM_OK	Statistics request successful
LCM_PRIM_NOK	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LRT_REASON_INVALID_INTERFACE	The specified interface does not exist (it is not configured)

dt

Date and time structure.

dura

Duration structure.

intfc

Interface for which statistics are requested. For further information, see Section 3.6.1.2.1, "**RtMiLrtCfgReq**."

s

```
typedef struct rtIntfcSts        /* Interface statistics */
{
   Cntr rtAttempt;        /* Number of route attempts */
   Cntr rtWrgCapTyp;      /* Failures because of wrong cap. type */
   Cntr rtAvail;          /* Number of routes when If was available */
   Cntr rtRoute;          /* Number of routes towards this interface */
   Cntr rtUnavail;        /* Failures because if was not available */
   Cntr rtCong1;          /* Number of routes during congestion 1 */
   Cntr rtCong2;          /* Number of routes during congestion 2 */
   Cntr rtCong3;          /* Number of routes during congestion 3 */
   Cntr rtPCong1;         /* Number of priority calls during cong 1 */
   Cntr rtPCong2;         /* Number of priority calls during cong 2 */
   Cntr rtPCong3;         /* Number of priority calls during cong 3 */
   Cntr rtReatCong1;      /* Failures because of congestion 1 */
   Cntr rtReatCong2;      /* Failures because of congestion 2 */
   Cntr rtReatCong3;      /* Failures because of congestion 3 */
} RtIntfcSts;
```

rtAttempt

Maintains the number of calls for which this interface was selected.

rtWrgCapTyp

Number of calls for which this interface is selected, yet, it cannot be used because the supported capability type does not match the capability type required for the connection.

rtAvail

Number of calls for which the interface was available.

rtRoute

Number of calls routed toward this interface.

rtUnavail

Number of calls for which this interface is not selected because it is not available.

rtCong1, rtCong2, rtCong3

Number of calls routed toward this interface during congestion levels 1, 2, and 3, respectively.

rtPCong1, rtPCong2, rtPCong3

Number of priority calls routed toward this interface during congestion levels 1, 2, and 3, respectively.

rtReatCong1, rtReatCong2, rtReatCong3

Number of calls for which this interface could not be selected due to congestion levels 1, 2, and 3, respectively.

Description:

The RT uses this function to provide the layer manager with statistics information at a particular interface.

Returns:

00 **ROK**

01 **RFAILED**

3.6.1.2.7 RtMiLrtStaReq

Name:

Status Request

Direction:

Layer manager to the RT

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RtMiLrtStaReq(pst, sta)
Pst      *pst;
RtMngmt  *sta;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sta

Pointer to the solicited status structure. The solicited status structure has the following format:

```
typedef struct rtMngmt
{
   Header hdr;
   CmStatus  cfm;                 /* status in confirm */
   union
   {
 /* solicited status */
     struct
      {
         DateTime dt;            /* date and time */
         union
         {
           RtRouteSta rtSta;     /* Route status */
           RtIntfcSta intfSta;   /* Interface status */
         } s;
      } ssta;                    /* solicited status */
   } t;
} RtMngmt;
```

hdr

Header structure. For this primitive, the **elmnt** field in the element ID (**elmId**) structure defines the element. For more details, see Section 3.3.1, "Header." The allowable values are:

Value	Description
LRT_ROUTE	Route
LRT_INTF	Interface

cfm

Status field. Only the confirmation primitives use this field to report errors. It is not significant to the status request. For further details, see Section 3.3.2, "Status."

dt

Date and time structure. It is previously described.

rtSta

Routes the status structure.

```
typedef  RtRoutCfg RtRouteSta;

typedef struct rtRoutCfg         /* Route Configuration */
{
   RtRoute     rt;               /* Routing info */
   U8          vpnId;            /* Identifier for the VPN */
   U8          routLoc;          /* location of the route
                                  * address
                                  */
   S16         nmbIntfc;         /* number of IF leading to a
                                  * specific rout */
   RmInterface intfc[LRT_MAXNUMINTF]; /* destination point code */
   U8          strpType;         /* type of digit stripping :
                                    route based,
                                    route + interface based */
   union
   {
      RtDgtsStrpInfo  rtDgtsStrpInfoS; /* simple method for stripping */
       RtDgtsStrpInfo  rtDgtsStrpInfoC[LRT_MAXNUMINTF];
                                       /* complete method */
   } rtDgtsStrp;
} RtRoutCfg;
```

rt

The **rt** field specifies the route for which status information is requested. The field is described in detail in the configuration request primitive.

The following are used only in the status confirm:

vpnId, routLoc, nmbIntfc, intfc, strpType, rtDgtsStrpInfoS, and **rtDgtsStrpInfoC.**

intfSta

Interface status structure. The **intfc** field specifies the interface for which the status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. See **RtMiLrtStaCfm** on page 164 for more information.

```
typedef struct rtIntfcSta    /* Interface status */
{
  U8          vpnId;         /* Identifier for the VPN */
  RmInterface intfc;         /* Interface */
  U16 nmbRoutes;             /* number of routes usingthis interface */
  U8  availSta;              /* availability of the interface */

} RtIntfcSta;
```

intfc

Interface for which status information is requested. The field is described in detail in the configuration request primitive. See Section 3.6.1.2.1, "**RtMiLrtCfgReq**."

vpnId, nmbRoutes, and **availSta**

They are used only in the status confirm.

Description:

The layer manager uses this function to gather solicited status information.

Returns:

00 ROK

01 RFAILED

3.6.1.2.8 RtMiLrtStaCfm

Name:

Status Confirm

Direction:

RT to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RtMiLrtStaCfm(pst, sta)
Pst     *pst;
RtMngmt *sta;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sta

Pointer to the solicited status structure. The solicited status structure has the following format:

```
typedef struct rtMngmt
{
   Header hdr;
   CmStatus  cfm;                  /* status in confirm */
   union
   {
 /* solicited status */
      struct
      {
         DateTime dt;              /* date and time */
         union
         {
           RtRouteSta rtSta;       /* Route status */
           RtIntfcSta intfSta;     /* Interface status */
         } s;
      } ssta;                      /* solicited status */
   } t;
} RtMngmt;
```

hdr

Header structure. For more details, see Section 3.3.1, "Header."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
   U16   status;              /* Status of the operation  */
   U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Statistics request successful
LCM_PRIM_NOK	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LRT_REASON_INVALID_INTERFACE	Specified interface does not exist (not configured)
LRT_REASON_INVALID_ROUTE	Specified route does not exist (not configured)

dt

Date and time structure.

rtSta

Route status structure.

```
typedef  RtRoutCfg RtRouteSta;

typedef struct rtRoutCfg        /* Route Configuration */
{
   RtRoute     rt;              /* Routing info */
   U8          vpnId;           /* Identifier for the VPN */
   U8          routLoc;         /* location of the route
                                 * address
                                 */
   S16         nmbIntfc;        /* number of IF leading to a
                                 * specific rout */
   RmInterface intfc[LRT_MAXNUMINTF]; /* destination point code */
   U8          strpType;        /* type of digit stripping :
                                   route based,
                                   route + interface based */
   union
   {
     RtDgtsStrpInfo  rtDgtsStrpInfoS; /* simple method for stripping */
      RtDgtsStrpInfo  rtDgtsStrpInfoC[LRT_MAXNUMINTF];
                                     /* complete method */
   } rtDgtsStrp;
} RtRoutCfg;
```

rt

Route for which status information is requested. The field is described in detail in the configuration request primitive. Refer to **rtRout** on page 132 for more details.

The following are used:

vpnId, routLoc, nmbIntfc, intfc, strpType, rtDgtsStrpInfoS, and **rtDgtsStrpInfoC.**

intfSta

Interface status structure.

```
typedef struct rtIntfcSta    /* Interface status */
{
  U8             vpnId;     /* Identifier for the VPN */
  RmInterface    intfc;     /* Interface */
  U16 nmbRoutes;           /* number of routes usingthis interface */
  U8  availSta;             /* availability of the interface */
} RtIntfcSta;
```

intfc

Interface for which status information is requested. The field is described in detail in the configuration request primitive. See Section 3.6.1.2.1, "**RtMiLrtCfgReq**."

vpnId

See **vpnId** on page 135 for more information.

nmbRoutes

This field specifies the number of routes using this interface.

availSta

The status of the interface. The possible values are:

Value	Description
CC_ME_INTFC_UNAVAIL	Available
CC_ME_INTFC_AVAIL	Unavailable (PAUSE)
CC_ME_INTFC_CONG1	Congested, level 1
CC_ME_INTFC_CONG2	Congested, level 2
CC_ME_INTFC_CONG3	Congested, level 3

Description:

The RT uses this function to return solicited status information to the layer manager.

Returns:

00 ROK

01 RFAILED

3.6.1.2.9 RtMiLrtStaInd

Name:

Status Indication

Direction:

RT to the layer manager

Supplied:

In the layer manager

Synopsis:

```
PUBLIC S16 RtMiLrtStaInd(pst, sta)
Pst     *pst;
RtMngmt *sta;
```

Parameters:

pst

For more information, see Section 3.3.3, "**Pst**."

sta

Pointer to the status structure. The status structure has the following format:

```
typedef struct rtMngmt
{
   Header hdr;
   CmStatus  cfm;                 /* status in confirm */
   union
   {
      /* unsolicited status */
      struct
      {
         CmAlarm  alarm;          /* alarm */
         union
         {
            SpId         spId;    /* service provider id */
            RmInterface  intfc;   /* Interface */
         }t;
      } usta;
   } t;
} RtMngmt;
```

hdr

For more description, see Section 3.3.1, "Header."

cfm

The status field is not significant to this primitive.

alarm

```
typedef struct cmAlarm
{
   DateTime dt;      /* data and time */
   U16 category;     /* alarm category*/
   U16 event;        /* alarm event */
   U16 cause;        /* alarm cause */
}CmAlarm;
```

dt

Date and time structure.

category

This field specifies the category to which the error is related. Currently, only one category is supported.

Name	Description
LCM_CATEGORY_INTERFACE	When an event is received on an SAP that is not configured nor bound

event

This field specifies the event that has occurred.

Name	Description
LCM_EVENT_INV_STATE	Invalid SAP state (SAP is not bound)
LCM_EVENT_UI_INV_EVT	Invalid event received from the upper layer
LRT_EVENT_PAPAUD_SEQ	Out-of-sequence for PAP auditing
LRT_EVENT_OAPAUD_SEQ	Out-of-sequence for OAP auditing
LRT_EVENT_PAPAUD_CFMTMR	Audit confirm timer expired for the PAP
LRT_EVENT_OAPAUD_CFMTMR	Audit confirm timer expired for the OAP
LRT_EVENT_PAPAUD_PEORIDTMR	Period timer for auditing has expired
LRT_EVENT_PAPAUD_FINISHED	PAP auditing finished
LRT_EVENT_OAPAUD_FINISHED	OAP auditing finished

cause

This field specifies the cause. The additional information in union **t** depends on the cause.

Name	Description
LCM_CAUSE_INV_SAP	Invalid SAP. The value that caused the problem is passed in the **spId** field.
LRT_CAUSE_INV_INTERFACE	An invalid interface was specified. The interface value that caused the problem is passed in the **interface** field.
LRT_CAUSE_AUD_CFM_OUTOFSEQENCE	Out-of-sequence for auditing
LRT_CAUSE_AUD_TMR_EXP	Auditing timer has expired
LRT_CAUSE_AUD_FINISHED	Auditing finished

spId

The **spId** in case of **LCM_CAUSE_INV_SAP**.

intfc

The interface in case of **LRT_CAUSE_INV_INTERFACE**.

Description:

The RT uses this function to provide the layer manager with unsolicited status information (alarms). The unsolicited status can be enabled or disabled via the layer manager control request. The RT generates the following alarms.

Description	Category	Event	Cause
Invalid SAP ID received in the RT primitives	**LCM_CATEGORY_INTERFACE**	**LCM_EVENT_UI_INV_EVT**	**LCM_CAUSE_INV_SAP**
State of the SAP associated with the RT primitive is not bound	**LCM_CATEGORY_INTERFACE**	**LCM_EVENT_INV_STATE**	**LCM_CAUSE_INV_SAP**
Interface provided in **RtUiRttMntStaReq** is not configured	**LCM_CATEGORY_INTERFACE**	**LCM_EVENT_UI_INV_EVT**	**LRT_CAUSE_INV_INTERFACE**
PAP has finished	**LCM_CATEGORY_PROTOCOL**	**LRT_EVENT_PAPAUD_FINISHED**	**LRT_CAUSE_AUD_FINISHED**
OAP has finished	**LCM_CATEGORY_PROTOCOL**	**LRT_EVENT_OAPAUD_FINISHED**	**LRT_CAUSE_AUD_FINISHED**
Audit confirm timer expires for the PAP	**LCM_CATEGORY_PROTOCOL**	**LRT_EVENT_PAPAUD_CFMTMR**	**LRT_CAUSE_AUD_TMR_EXP**

Description	Category	Event	Cause
Audit confirm timer expires for the OAP	**LCM_CATEGORY_PROTOCOL**	**LRT_EVENT_OAPAUD_CFMTMR**	**LRT_CAUSE_AUD_TMR_EXP**
Period timer expires for the PAP	**LCM_CATEGORY_PROTOCOL**	**LRT_EVENT_PAPAUD_PERIODTMR**	**LRT_CAUSE_AUD_TMR_EXP**
PAP audit confirm is out-of-sequence	**LCM_CATEGORY_PROTOCOL**	**LRT_EVENT_PAPAUD_REQ**	**LRT_CAUSE_AUD_CFM_OUTOFSEQENCE**
OAP audit confirm is out-of-sequence	**LCM_CATEGORY_PROTOCOL**	**LRT_EVENT_OAPAUD_REQ**	**LRT_CAUSE_AUD_CFM_OUTOFSEQENCE**

Returns:

00 **ROK**

01 **RFAILED**

3.6.2 Interface with the Upper Layers

The RT provides routing functionality to GCC. The following primitives are provided at this interface, which is called the RTT interface.

Bind Establishment

Primitive Name	Description	Flow
`XxYyRttBndReq`	Bind request	GCC to RT
`XxYyRttBndCfm`	Bind confirm	RT to GCC

Route Determination

Primitive Name	Description	Flow
`XxYyRttRteReq`	Route request	GCC to RT
`XxYyRttRteCfm`	Route confirm	RT to GCC
`XxYyRttRteRsp`	Route response	GCC to RT
`XxYyRttRelReq`	Release request	GCC to RT
`XxYyRttRelInd`	Release request	RT to GCC
`XxYyRttMntStaReq`	Maintenance status request	GCC to RT
`XxYyRttMntStaInd`	Maintenance status indication	RT to GCC

For a detailed description of the RTT interface, refer to the *Interworking Call Control Interface Service Definition*.

3.6.3 Interface with System Services

This section discusses RT’s interface with system services.

3.6.3.1 General

This section describes system services required by the RT.

Task Scheduling

The task scheduling management functions are called to register, activate, and terminate a task. Use the following functions for task scheduling management.

Name	Description
`SRegActvTsk`	Registers an activate task - Task
`rtActvTsk`	Activates task for the RT
`SPstTsk`	Posts a task
`SExitTsk`	Exits a task

Initialization

OS calls the initialization management function to initialize a task. Use the following function for initialization management.

Name	Description
`rtActvInit`	Activates a task - Initialize the RT

Memory Management

The memory management functions allocate and deallocate variable-sized buffers using static buffers. Use the following functions for memory management.

Name	Description
`SGetSBuf`	Gets static buffer
`SGetSMem`	Gets static memory

Message Management

The message management functions initialize, add data to, and remove data from messages utilizing dynamic buffers. Use the following functions for message management.

Name	Description
SGetMsg	Allocates a message (from a dynamic pool)
SPutMsg	Deallocates a message (into a dynamic pool)
SInitMsg	Initializes a message
SFndLenMsg	Finds the length of a message
SExamMsg	Examines an octet at a specified index in a message
SAddPreMsg	Adds an octet to the beginning of a message
SAddPstMsg	Adds an octet to the end of a message
SRemPreMsg	Removes an octet from the beginning of a message
SRemPstMsg	Removes an octet from the end of a message
SPkS8	Adds a signed 8-bit value to a message
SPkU8	Adds an unsigned 8-bit value to a message
SPkS16	Adds a signed 16-bit value to a message
SPkU16	Adds an unsigned 16-bit value to a message
SPkS32	Adds a signed 32-bit value to a message
SPkU32	Adds an unsigned 32-bit value to a message
SUnpkS8	Removes a signed 8-bit value from a message
SUnpkU8	Removes an unsigned 8-bit value from a message
SUnpkS16	Removes a signed 16-bit value from a message
SUnpkU16	Removes an unsigned 16-bit value from a message
SUnpkS32	Removes a signed 32-bit value from a message
SUnpkU32	Removes an unsigned 32-bit value from a message

Miscellaneous

Resource availability checking. The following miscellaneous functions are used.

Name	Description
`SFndProcId`	Finds a processor ID on which a task runs
`SGetDateTime`	Gets the real date and time
`SLogError`	Handles an error
`SPrint`	Prints a preformatted string to default a display device

For a detailed description of the system services listed previously, refer to the *System Services Interface Service Definition*.

3.6.3.2 rtActvInit

Name:

Activate Task - Initialize the RT

Direction:

System services to the RT

Supplied:

Yes

Synopsis:

```
PUBLIC S16 rtActvInit(ent, inst, region, reason)
Ent    ent;
Inst   inst;
Region region;
Reason reason;
```

Parameters:

ent

Entity ID.

inst

Instance ID for the entity.

region

Memory region ID that may be used by the layer to get static memory.

reason

Reason for initialization. Currently, this field is not used.

Description:

System services uses this function to initialize the RT. The pointer to this function is passed to system services when registering the task.

Returns:

00 ROK

01 RFAILED

3.6.3.3 rtActvTsk

Name:

Activate Task

Direction:

System services to the RT

Supplied:

Yes

Synopsis:

```
PUBLIC S16 rtActvTsk(pst, mBuf)
Pst    *pst;
Buffer *mBuf;
```

Parameters:

pst

For more information, see Section 3.3.3, "**Pst**."

mBuf

Message buffer.

Description:

System services uses this function, which injects an event and primitive into the RT layer. The given message buffer is unpacked to find the corresponding primitive and associated parameters. Then, the appropriate primitive reception handler is scheduled.

Returns:

00 **ROK**

01 **RFAILED**

3.7 Resource Manager

This section discusses the Resource Manager (RM), detailing its interfaces and associated primitives.

3.7.1 Interface with the Layer Manager

This section discusses RM's interface with its layer manager.

3.7.1.1 Primitive Overview

The following is a list of primitives used between the RM and its layer manager (LRM).

Configuration

The following primitives configure protocol layer resources.

Name	Description
`RmMiLrmCfgReq`	Configuration request
`RmMiLrmCfgCfm`	Configuration confirm

Control

The following primitives control the RM.

Name	Description
`RmMiLrmCntrlReq`	Control request
`RmMiLrmCntrlCfm`	Control confirm

Statistics

The following primitives retrieve statistics information.

Name	Description
`RmMiLrmStsReq`	Statistics request
`RmMiLrmStsCfm`	Statistics confirm

Solicited Status

The following primitives retrieve the status of internal RM information.

Name	Description
RmMiLrmStaReq	Status request
RmMiLrmStaCfm	Status confirm

Unsolicited Status

The RM uses the following function to indicate status changes.

Name	Description
RmMiLrmStaInd	Status indication

3.7.1.2 Specific

This section describes the primitives passed at the interface between the RM and its layer manager, which is called the RMT interface.

3.7.1.2.1 RmMiLrmCfgReq

Name:

Configuration Request

Direction:

Layer manager to the RM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RmMiLrmCfgReq(pst, cfg)
Pst     *pst;
RmMngmt *cfg;
```

Parameters:

`pst`

For more description, see Section 3.3.3, "**`Pst`**."

cfg

Pointer to the configuration structure. The configuration structure has the following format:

```
typedef struct rmMngmt
{
   Header    hdr;                  /* header */
   CmStatus  cfm;                  /* status in confirm */
   union
   {
/* Configuration */
      struct
      {
         union
         {
             RmGenCfg   rmGen;      /* Resource Management General Config */
            RmSapCfg   rmSapCfg;   /* Upper Sap Configuration */
            RmBbPhyCfg rmBbPhyCfg; /* Broadband Physical Link
                                       Configuration */
            RmBbIntfcCfg rmBbIntfcCfg; /* Broadband Interface
                                           Configuration */
            RmNbDpcCfg rmNbDpcCfg; /* Narrowband DPC Configuration */
            RmVpCfg    rmVpCfg;    /* VPCI Configuration */
            RmCicCfg   rmCicCfg;   /* CIC Configuartion */
             RmPvcCfg   rmPvcCfg;   /* Static Binding (PVC) Configuration */
            RmDss1IntfcCfg rmDss1IntfcCfg;
                                   /* DSS1 Interface Configuration */

            /* Following field is significant in case of CfgCfm */
            RmDiag     diagn;      /* Diagnostics */
         }s;
      } cfg;
```

hdr

Header structure. The **elmnt** field in the element ID (**elmId**) structure defines the element. For more information, see Section 3.3.1, "Header."

The allowable values for this primitive are:

Value	Description
STRMUPSAP	Call control SAP element
STRMBBPHY	Broadband physical link
STRMNBDPC	Narrowband DPC
STRMVP	VPCI
STRMCIC	CIC
STRMPVC	PVC (static binding)
STRMVC	VC
STRMDSS1INTFC	DSS1 interface
STRMBBINTFC	Broadband interface

cfm

Status field. The confirmation primitives use this field to report errors. It is not significant to the configuration request. See Section 3.3.2, "Status."

rmGen

General RM configuration structure. The general configuration must be done first. The RM uses much of the information in this table to reserve the proper amount of static memory.

```
typedef struct rmGenCfg    /* General Configuration of Resource Manager
                            */
{
   U8  maxSap;             /* Maximum Number of CC and PSIF SAP's */
   U8  maxBbDpc;           /* Maximum Number of BB DPC's Configured */
   U16 maxBbPhy;           /* Maximum Number of BB Physical Links Cfg */
   U8  maxNbDpc;           /* Maximum Number of NB DPC's Configured */
   U32 maxVp;              /* Maximum Number of BB VPCI in System */
   U32 maxVc;              /* Maximum Number of BB VPCI/VCI in System */
   U32 maxCic;             /* Maximum Number of NB CIC Cfg in System */
   U32 maxPvc;             /* Maximum Number of PVC's in the System */
   U16 vpTblHlSz;          /* VP Table Hash List Size */
   U16 vcTblHlSz;          /* VC Table Hash List Size */
   U32 maxDss1Intfc;       /* Maximum Number of DSS1 Interfaces */
   U32 maxDss1PriLnk;      /* Maximum Number of PRI Links */
   U16 dss1TblHlSz;        /* Size of DSS1 Interface Hash Table */
   U32 maxBbIntfc;         /* Maximum Number of ATM UNI/PNNI */
   U16 bbIntfcTblHlSz;     /* Size of ATM UNI/PNNI Intfc Hash Table */
   Pst sm;                 /* Post Structure to Stack Manager */
#ifdef ICC_AUDIT
   S16 timeRes;            /* time resolution */
#endif /* ICC_AUDIT */
} RmGenCfg;
```

maxSap

Number of SAPs. This is the maximum number of SAPs toward GCC and the PSIFs. The allowable values are: 1 to 255.

maxBbDpc

Maximum number of broadband DPCs. The allowable values are: 1 to 255.

maxBbPhy

Maximum number of broadband physical links configured in the system. The allowable values are: 1 to 32767.

maxNbDpc

Maximum number of narrowband DPCs. The allowable values are: 1 to 255.

maxVp

Maximum number of VPIs.

maxVc

Maximum number of VPI/VCIs on which a call may be active at any time.

maxCic

Maximum number of narrowband resources (circuits) configured.

Note: *For the RM, the maximum number of narrowband resources corresponds to the sum of the highest numbered CIC assigned on each DPC. Some of the CICs may not actually be configured.*

maxPvc

Maximum number of static bound resources. Static binding means that two network resources on different interfaces are bound together, so that a call incoming on one resource *must* be routed on the second resource, which is identified in the binding. The association between the incoming and outgoing resource is predefined.

In case of ISUP-to-B-ISUP interworking, static binding means that each CIC is associated with a VPI/VCI. Each call on that particular CIC must go to the predefined VPI/VCI. Outgoing resources cannot be used. The resource cannot be part of a different call when it is used as part of a static binding.

Static binding is used when the interworking unit does not have a switching fabric. There is a direct physical connection between the CICs and VPI/VCIs. The interworking unit maps the signalling information from one protocol to the other, but does not perform any bearer-channel switching.

vpTblHlSz

Size of the VPI hash list. The ideal value equals the number of parallel connections that can exist in the system. In this case, each hash list bin has a maximum of one entry and the search time is minimal. Reducing the size of the hash list increases the search time, but the memory requirement is less. There is always a trade-off between time and memory. A good value is about one-fourth the number of connections, so that a hash list bin has a maximum of four entries.

vcTblHlSz

VPI/VCI hash table size. See **vpTblHlSz** (above) on how to choose the value.

maxDss1Intfc

Maximum number of DSS1 interfaces. The allowable value: 32-bit integer.

maxDss1PriLnk

Maximum number of Primary Rate Interface (PRI) links. This is different from **maxDSS1Intfc** since a DSS1 interface may have multiple-PRI access. The allowable value: 32-bit integer.

dss1TblHlSz

Maximum number of bins for the DSS1 interface hash table. See **vpTblHlSz on page 184** on how to choose the value.

maxBbIntfc

Maximum number of broadband interfaces. The allowable value: 32-bit integer.

bbIntfcTblHlSz

Maximum number of bins for the broadband interface hash table. See **vpTblHlSz on page 184** on how to choose the value.

sm

Post structure. It is used for communicating with the stack manager. The RM requires the post structure when sending an unsolicited status. An unsolicited status is sent to the address in the **sm** field.

timeRes

Timer resolution, that is, the period during which the common timer function is called for this module. The module uses this period internally to maintain different timers for different connections.

rmSapCfg

Upper SAP configuration structure. This SAP is used to communicate with GCC and the PSIF.

```
typedef struct rmSapCfg         /* SAP Configuration structure */
{
   SpId      spId;              /* Sap Being Configured */
   Priority  prior;             /* Priority */
   Selector  selector;          /* Selector */
   Route     route;             /* Route */
   MemoryId  mem;               /* Memory Region & Pool Id */
#ifdef ICC_AUDIT
   U8    cid;
   U8    maxNumAuditRsc;        /* maxmum number of resource can be
                                  audited through one audit requests */

   RmSapTmrCfg tmr;
#endif /* ICC_AUDIT */
} RmSapCfg;
```

spId

Service provider ID. The RM uses this **spId** to identify the SAP on which it communicates with GCC or the PSIF.

prior

Priority. It is used when the task buffers must be posted between the service provider and service user. It is used only in a loosely coupled system. The allowable value is:

PRIOR0 priority 0 - highest

selector

Defines whether the service provider and service user are loosely or tightly coupled. It is used to resolve a primitive call to the upper layer. The allowable values depend on the configuration. Refer to the *ICC Portation Guide* for more details.

route

The system uses this for internal routing requirements. TAPA does not define the contents or use of this information. It is used only in a loosely coupled system. The allowable value is:

RTESPEC route to the specific instance

mem

For more description, see Section 3.3.5, "**Memory**."

cid

ID of the GCC entity, which binds to this SAP.

maxNumAuditRsc

Maximum number resource audited in an audit request.

tmr

SAP **timres**, which is defined as:

```
typedef struct RmSapTmrCfg
{
   TmrCfg audCfmTmr;       /* timer for waiting for Audit Confirm
                              for PAP */
   TmrCfg periodAudTmr;    /* timer Periodical Auditing*/
} RmSapTmrCfg;
```

audCfmTmr

The time the RM waits to receive the audit confirm for the pending audit request. For further details, refer to Section 3.3.4, "Timer Configuration."

periodAudTmr

Periodic auditing timer. For more details, refer to Section 3.3.4, "Timer Configuration."

rmBbPhyCfg

Broadband physical interface configuration.

```
typedef struct rmBbPhyCfg      /* Broadband Physical Link
                                  Configuration */
{
   U32  intfcId;               /* Physical Interface Id */
   RmTfcMtrc  defMtrc[MAX_QOS_CLASSES]; /* Maximum Forward Cell Rate 
capacity */
   RmTfcThreshold tfcThreshold[MAX_QOS_CLASSES]; /* Maximum Backward 
Cell Rate capacity */
} RmBbPhyCfg;
```

intfcId

Interface ID.

defMtrc

Default traffic metric parameters configured for this physical link. The traffic metric parameters are maintained per QoS service class. The following service classes are possible:

Class	Description
PN_QOS_CLASS_UBR	Unspecified
PN_QOS_CLASS_CBR	Constant bit rate
PN_QOS_CLASS_VBR_RT	Variable bit rate—real time
PN_QOS_CLASS_VBR_NRT	Variable bit rate—**n** real time
PN_QOS_CLASS_ABR	Available bit rate

```
typedef struct rmTfcMtrc
{
   U32 clp;             /* GCAC Cell Loss Priority bit */
   U32 adminWt;         /* administrative weight */
   U32 maxFCR;          /* max forward cell rate (cells/sec) */
   U32 maxBCR;          /* max backward cell rate (cells/sec) */
   U32 ctd;             /* cell transfer delay (10 u-sec) */
   U32 cdv;             /* cell delay variation (10 u-sec) */
   U16 clr0;            /* cell loss ratio CLP = 0 */
   U16 clr1;            /* cell loss ratio CLP = 1 */
   U32 crm;             /* opt GCAC : cell rate margin */
   U32 vf;              /* opt GCAC : variance factor (2**-8) */
   U16 tmFlgs;          /* GCAC parameters present */
} RmTfcMtrc;
```

clp

GCAC cell loss priority bit. The allowable values are 0/1.

adminWt

Administrative weight. Default 5040, additive.

maxFCR

Maximum forward cell rate supported by this link, in cells/sec.

maxBCR

Maximum backward cell rate supported by this link, in cells/sec.

ctd

Cell transmission delay in units of ten microseconds.

cdv

Cell delay variation in units of ten microseconds.

clr0

Cell loss ratio when the cell loss priority is 0.

clr1

Cell loss ratio when the cell loss priority is 0+1.

crm

Cell rate margin is an optional GCAC parameter.

vf

Variance factor is an optional GCAC parameter.

tmFlgs

Optional generic call admission control parameters are present. The allowable values are:

Value	Description
PN_TM_FLG_GCAC_PRES	Present
NOTPRSNT	Not present

Note: *For broadband links between the B-ISUP nodes, only the* **MAXFCR** *and* **maxBCR** *parameters are relevant.*

tfcThreshold

Thresholds for various traffic metric parameters configured for this physical link. The RM updates the traffic metric parameters on a per call basis and generates an **RmMiLrmStaInd** toward the layer manager, if the threshold for any traffic metric parameter is reached. A threshold value of zero indicates that **RmMiLrmStaInd** does not need to be generated for that parameter.

```
typedef struct rmTfcThreshold
{
   U8 avCRThreshold; /* Threshold for available cell rate expressed
in  % */
} RmTfcThreshold;
```

avCRTheshold

Threshold for available cell rate expressed as a percentage of **maxFCR**.

rmBbIntfcCfg

Broadband interface configuration.

```
typedef struct rmBbIntfcCfg      /* Broadband Interface Configuration */
{
   RmInterface bbIntfc;          /* Broadband Interface */
   U8   alocMeth;                /* Allocation Method to be used */
#ifdef ICC_AUDIT
   SpId sapId;                   /* PS SAP this interface belongs to */
#endif /* ICC_AUDIT */
} RmBbIntfcCfg;
```

bbIntfc

Broadband interface. It is defined as:

```
typedef struct rmInterface    /* Generic Interface Structure */
{
   U8  intfType;              /* Identifies the Interface type */
   union interface
   {
     Dpc   dpc;               /* For ISUP, BISUP Interface type */
     U32   intfcId;           /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The allowable values are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP node.

intfcId

Interface ID. The allowable value: 32-bit integer.

alocMeth

The method by which the broadband resource should be allocated. Currently, only one method, the lowest available VPI/VCI, is available. The allowable value is:

LRM_AM_LOWEST_AVAIL **Lowest Available**

sapId

The SAP ID with which this interface associates.

rmNbDpcCfg

Narrowband DPC configuration.

```
typedef struct rmNbDpcCfg    /* Narrowband DPC Configuration */
{
   Dpc  dpc;                 /* DPC to be configured */
   U8   alocMeth;            /* Allocation Method to be used */
   U16  maxCic;              /* Highest CIC configured on this DPC */
   U32  cotFrequency;        /* Continuity check allowed */
#ifdef ICC_AUDIT
   SpId sapId;               /* PS SAP this DPC belongs to */
#endif /* ICC_AUDIT */
} RmNbDpcCfg;
```

dpc

Destination point code.

alocMeth

The method by which the ISUP resource (circuits) should be allocated. The following methods are available:

- Highest available: The highest available CIC is selected.
- Lowest available: The lowest available CIC is selected.
- ITU-T method 2 (Q.764). As specified in Q.764, each node of a bothway circuit group has priority access to the group of circuits that it controls. Each node controls one half of the circuits in a bothway circuit group. The node with the higher signalling point code controls all the even-numbered circuits (CIC) and the other node controls the odd-numbered circuits.

  Within each group, the circuit that has been released the longest is selected (first-in, first-out). Each node of a bothway circuit group has non-priority access to the group of circuits that it does not control. Of this group, the latest released circuit is selected (last-in, first-out) if all the circuits in the group are busy.

  The allowable values are:

Value	Description
LRM_AM_LOWEST_AVAIL	Lowest CIC available
LRM_AM_HIGHEST_AVAIL	Highest CIC available
LRM_AM_ITU_MTHD2	ITU method 2 allocation

maxCic

The highest available CIC configured on this DPC.

cotFrequency

Continuity check frequency. It specifies the frequency with which the statistical continuity check is initiated at a particular interface.

sapId

SAP ID with which this DPC associates.

rmVpCfg

VPI configuration.

```
typedef struct rmVpCfg      /* Virtual Path Connection Configuration */
{
   RmInterface bbIntfc;     /* Broadband Interface */
   VpId  vpId;              /* VPCI */
   VcId  minVcId;           /* Minimum Valid VCI Value on this VPCI */
   VcId  maxVcId;           /* Maximum Valid VCI Value on this VPCI */
   Bool  isItAssg;          /* VPCI Assignability */
   U32   phyLnkId;          /* Physical Link Identifier */
   U32   maxFCR[MAX_QOS_CLASSES]; /* Array of Maximum Forward Cell Rate
                                  */
   U32   maxBCR[MAX_QOS_CLASSES]; /* Maximum Backward Cell Rate */
   U8    initState;               /* initial State of VPCI */
   U8   rmAffinity;               /* Resource Manager Affinity */
} RmVpCfg;
```

bbIntfc

Broadband interface. It has the following format.

```
typedef struct rmInterface    /* Generic Interface Structure */
{
   U8  intfType;              /* Identifies the Interface type */
   union interface
   {
       Dpc   dpc;             /* For ISUP, BISUP Interface type */
       U32   intfcId;         /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The allowable values are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP node.

intfcId

Interface ID. The allowable value: 32-bit integer.

vpId

The ID of the configured VP.

minVcid, maxVcid

These values identify the lowest and highest value of the VCIs used on this VPCI. All the values between the specified range are valid.

isItAssg

Identifies whether this node can assign VCIs on the indicated VPCI. The RM allocates bearer-channels on an assigning VPCI. The non-assigning VPCIs are rewired to be configured to do resource validation in the RM.

phyLnkId

These identify the physical link with which this VPCI is associated.

maxFCR, maxBCR

Maximum bandwidth allocated to this VPCI. The bandwidth allocable to each VPCI may be less than or equal to the total bandwidth available on the physical link. The maximum values allocable in the forward and backward directions may be different. The maximum bandwidth is maintained per QoS service class. The following service classes are possible.

Value	Description
PN_QOS_CLASS_UBR	Unspecified
PN_QOS_CLASS_CBR	Constant bit rate
PN_QOS_CLASS_VBR_RT	Variable bit rate—real time
PN_QOS_CLASS_VBR_NRT	Variable bit rate—**n** real time
PN_QOS_CLASS_ABR	Available bit rate

initState

Initial state of the VPI. The initial state of a VPI can be available (**TRUE**) or not available (**FALSE**) for allocation.

rmAffinity

RM affinity. It indicates whether allocating this resource contributes in allocating the other resource. The allowable values are:

Value	Description
RM_NOPREF	No preference
RM_STATICBND	Statically bound resource
RM_PREFERRED_RM	Second resource allocation depends on the first

rmCicCfg

Circuit configuration.

```
typedef struct rmCicCfg   /* Narrowband CIC Range Configuration */
{
   Dpc  dpc;               /* Narrowband DPC interface identification */
   U16  strtCic;          /* Starting CIC */
   U16  numCic;           /* number of CIC Configured */
   U8   cicType;          /* type of CIC */
   U8   cntld;            /* Which CICs are Controlled by other node */
   U8   initState;        /* Initial state of the CIC */
   U8   mgId;             /* Media Gateway Id */
   Bool viaSatellite;     /* The circuit is over satellite */
   Ticks delayVal;        /* Delay over this circuit */
   U8   cotChkFlag;       /* Continuity check allowed */
   U8   echoCntrlFlag;    /* Echo control allowed */
   U8   rmAffinity;       /* Resource Manager Affinity */
} RmCicCfg;
```

dpc

DPC. Identifies the interface at which the group of CICs are configured.

strtCic

Starting CIC configured on this DPC.

numCic

Number of CICs configured, starting from the CIC specified in the **strtCic**.

cicType

Identifies whether the CICs can be used for incoming/outgoing, or bothway calls. The allowable values are:

Value	Description
LRM_CIC_OUTGOING	Outgoing calls allowed
LRM_CIC_INCOMING	Incoming calls allowed
LRM_CIC_BOTHWAY	Bothway = INCOMING|OUTGOING

cntld

Specifies which circuits are controlled by the remote node. This field is used for ITU method 2 allocation. By default, all circuits are assumed to be in the controlling list of this node. This field contains a bit mask. The least significant bit (LSB, Bit 0) indicates that the odd circuits are controlled by the remote node. Bit 1 indicates that the even circuits are controlled by the remote node. If all the circuits are controlled by the remote node, both flags must be set. The allowable values are:

Value	Description
LRM_CNTRLD_ODD	Odd circuits are controlled
LRM_CNTRLD_EVEN	Even circuits are controlled
LRM_CNTRLD_ALL	All circuits are controlled (ODD|EVEN)

initState

Initial state of the circuit. The initial state of a circuit can be available (**TRUE**) or not available (**FALSE**) for allocation.

mgId

Media gateway ID. The RM uses this while allocating resources for a call from the same media gateway.

viaSatellite

Specifies that the circuit is over a satellite hop.

delayVal

Delay over this circuit. It specifies the delay values in milliseconds.

cotFrequency

Continuity check frequency. It specifies the frequency with which the statistical continuity check is initiated at a particular interface.

cotChkFlag

Continuity check indicator. It specifies whether the continuity check is enabled. The allowable values are:

Value	Description
RM_COT_NOK	Continuity check disabled
RM_IN_COT_OK	Incoming continuity check enabled
RM_OUT_COT_OK	Outgoing continuity check enabled

echoCntrlFlag

Echo control indicator. It specifies whether the echo control is enabled.

Value	Description
RM_ECHOCNTRL_NOK	Echo control is disabled
RM_IN_ECHOCNTRL_OK	Incoming echo control is enabled
RM_OUT_ECHOCNTRL_OK	Outgoing echo control is enabled

rmAffinity

RM affinity. This indicates whether allocating this resource contributes in allocating the other resource.

Value	Description
RM_NOPREF	No preference
RM_STATICBND	Statically bound resource
RM_PREFERRED_RM	Second resource allocation depends on the first

rmPvcCfg

Circuit configuration.

```
typedef struct rmPvcCfg        /* Resource Static Binding (PVC)
                                  Configuration */
{
   RmRsc  rsrc;                /* Resource */
   RmRsc  assocRsrc;           /* Associated Resurce */
} RmPvcCfg;
```

rsrc

Resource for which static binding is defined.

```
typedef struct rmRsc              /* Generic Resource Structure */
{
   RmInterface intfc;             /* Interface on which resource
                                     is identified */
   Bool  rscPres;                 /* True if the Resource has been
                                     Identified */
   union rsc
   {
       RmBbRsc    bbRsc;          /* Broadband Resource */
       RmNbRsc    nbRsc;          /* Narrowband Resource */
       RmDss1Rsc  dss1Rsc;        /* DSS1 Resource */
      RmBbPh1TrnkRsc bbPh1TrnkRsc; /* ATM phase1 trunking Resource
                                     */
      RmBbPh2TrnkRsc bbPh2TrnkRsc; /* ATM phase2 trunking Resource
                                     */
      RmFeatTrpRsc   featTrpRsc;   /* Feature transparency Resource
                                    */
       RmAtmTrnkRsc   atmTrnkRsc; /* ATM AAL1/AAL2 trunk Resource
                                    */
   }t;
}RmRsc;
```

intfc

Interface to which the resource belongs, either ISUP or B-ISUP. The allowable values are:

CC_BI_INTFC
CC_SI_INTFC

rscPres

This field identifies whether the following resource information (**bbRsc** or **nbRsc**) is valid. This must always be set to **TRUE** for the configuration request.

bbRsc

Broadband resource. This field must be filled when **intfc** identifies a B-ISUP interface type. This structure contains the VPI and VCI of the broadband resource. The flag field is insignificant for the B-ISUP interfaces.

```
typedef struct rmBbRsc            /* Broadband Resource */
{
   U8    flag;                    /* Flag */
   VpId  vpId;                    /* VPI */
   VcId  vcId;                    /* VCI */
} RmBbRsc;
```

nbRsc

Narrowband resource. This field must be filled when **intfc** identifies an ISUP interface type. This structure contains the CIC value of the narrowband resource.

```
typedef struct  rmNbRsc      /* Narrowband Resource */
{
   Cic   cic;                /* Circuit Identification Code */
} RmNbRsc;
```

dss1Rsc

DSS1 resource. This field must be filled when **intfc** identifies a DSS1 interface type. This structure contains the channel value of the DSS1 resource. Refer to the *INT Interface Service Definition* for more details.

```
typedef struct rmDss1Rsc    /* channel id tokens */
{
   ElmtHdr eh;              /* element header */
   TknU8   infoChanSel;     /* information channel selection */
   TknU8   dChanInd;        /* d channel indicator */
   TknU8   prefExc;         /* preferred/exclusive */
   TknU8   intType;         /* interface type */
   TknU8   intIdentPres;    /* interface identifier present */
   TknU16  intIdent;        /* interface identifier */
   TknU8   chanMapType;     /* channel type/map type */
   TknU8   nmbMap;          /* number/map */
   TknU8   codeStand1;      /* coding standard */
   TknStrM chanNmbSlotMap;  /* channel number/slot map */
} RmDss1Rsc;
```

bbPh1TrnkRsc, bbPh2TrnkRsc, featTrpRsc, and **atmTrnkRsc**

These resources are not configured in the RM. The XM dynamically creates and allocates these trunking resources.

assocRsrc

Resource statically bound to the resource identified above.

rmDss1IntfcCfg

Dss1 interface configuration.

```
typedef struct rmDss1IntfcCfg
{
   U32 intfcId;          /* DSS1 Interface Identifier */
   U8  rmtLclAloc;       /* Resource to be aloc by RM or Peer
                          * (or Q.931 layer) */
   U8 accessType;        /* BRI, PRI or NFAS */
   U8 alocMeth;          /* Allocation Method in case of
                          * PRI and NFAS */
   U8 chnl[LRMMAXPRICHNL]; /* Channels being Equipped for
                            * DSS1 Interface */
   U8 intId;             /* Interface Id Required in case of NFAS */
   U8   mgId;            /* Media Gateway Id */
   U8   rmAffinity;      /* Resource Manager Affinity */
#ifdef ICC_AUDIT
   SpId sapId;           /* PS SAP this interface belongs to */
#endif /* ICC_AUDIT */
} RmDss1IntfcCfg;
```

intfcId

The configured DSS1 interface ID.

rmtLclAloc

Indicates whether the RM or Q.931 layer allocates the resources for this interface. The allowable values are:

Value	Description
LRM_DSS1_LCL_ALOC	The RM should allocate the resources
LRM_DSS1_PEER_ALOC	The PEER or DSS1 should allocate the resources

accessType

This field indicates the type of access used—PRI, Basic Rate Interface (BRI), or multiple-PRI. The allowable values are:

Value	Description
LRM_BRI	Basic rate access
LRM_PRI	Primary rate access
LRM_NFAS	Non-Facility Associated (NFAS) access

alocMeth

The resource allocation method used for this interface. The following methods are available.

Name	Description
LRM_AM_HIGHEST_AVAIL	The highest available channel is selected
LRM_AM_LOWEST_AVAIL	The lowest available channel is selected

chnl

The list containing the initial state of DSS1 channels equipped for the DSS1 interface. A DSS1 channel can be configured with one of the following initial states.

Value	Description
LRM_IDLE	Resource is IDLE
LRM_UNEQUIP	Resource is unequipped
LRM_CP_BSY	Resource is CP busy
LRM_MNT_BSY	Resource is maintenance busy

intId

The interface ID for the PRI link in the DSS1 interface. This parameter is valid if a multiple-PRI access is used for the DSS1 interface.
The allowable values are: 0 to 255.

mgId

Media gateway ID. The RM uses this while allocating resources for a call from the same media gateway.

rmAffinity

RM affinity. This indicates whether allocating this resource contributes in allocating the other resource. The allowable values are:

Value	Description
RM_NOPREF	No preference
RM_STATICBND	Statically bound resource
RM_PREFERRED_RM	Second resource allocation depends on the first

`sapId`

SAP ID of this interface.

`diagn`

This field is significant only to the configuration confirm primitive. For more information, see Section 3.7.1.2.2, "**`RmMiLrmCfgCfm`**."

Description:

The layer manager uses this function to configure the RM. General configuration must be done first. The interface (DPC) must be configured before the resources (VPI, circuits) and the static binding can be configured.

The following configuration order is suggested:

1. General
2. Interface (DPC configuration), narrowband DPC, broadband interface, and DSS1 interface
3. Broadband physical link
4. Resource, VPI, and circuit
5. Static binding (if required)

Returns:

00 ROK

01 RFAILED

3.7.1.2.2 RmMiLrmCfgCfm

Name:

Configuration Confirm

Direction:

RM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RmMiLrmCfgCfm(pst, cfg)
Pst     *pst;
RmMngmt *cfg;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. With the exception of the **status**, **reason**, and **diagn** fields described next, the structure used for configuration confirm is the same as that for the configuration request. See Section 3.7.1.2.1, "**RmMiLrmCfgReq**."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
    U16   status;              /* Status of the operation  */
    U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous configuration request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_ELMNT	Invalid element specified in the configuration request
LCM_REASON_RECONFIG_FAIL	Failure in reconfiguration
LCM_REASON_MEM_NOAVAIL	Memory allocation failed
LCM_REASON_GENCFG_NOT_DONE	Configuration request received without any previous general configuration
LCM_REASON_INVALID_SAP	Invalid SAP value passed. The passed SAP does not exist in the system.
LCM_REASON_EXCEED_CONF_VAL	Configuration requests exceed the maximum value as passed in the general configuration. For example, more DPCs configured than the maximum specified in the general configuration.
LCM_REASON_HASHING_FAILED	Hashing library returned failure
LCM_REASON_INVALID_PAR_VAL	One of the passed parameters is invalid. For example, the DPC passed in the VPI configuration does not exist—that is, it was not configured prior to the VPI configuration.

diagn

Provides further information about a given error. This field is significant only when the status indicates that the request failed.

```
typedef struct rmDiag
{
   union
   {
     struct
     {
       U32 parId;      /* Paramter Identifier */
       RmRsc rsc;      /* Resource */
     }s;
#ifdef PNNI_ROUTING_ENABLED
     PnMtrcCfg pnMtrcCfg; /* traffic matrix */
#endif
   }t;
} RmDiag;
```

parId

Describes the parameter causing the problem. The following values are possible.

Value	Description
LRM_PPHY	Physical link
LRM_PUPSAP	Resource manager SAP
LRM_PBBDPC	Parameter type BB DPC
LRM_PVP	Parameter type VP
LRM_PNBDPC	Parameter type NB DPC
LRM_PCIC	Parameter type CIC
LRM_PRSC	Parameter type resource
LRM_PINTID	Parameter interface ID, in case of NFAS
LRM_PINTFCID	Parameter DSS1 interface ID
LRM_PDSS1PRILNK	Parameter DSS1 PRI link
LRM_PBBINTFC	Parameter broadband interface
LRM_PVC	Parameter type VC
LRM_PINTFCTYPE	Parameter interface type
LRM_POBJTYPE	Parameter object type

rsc

The interface or resource causing the error.

pnMtrcCfg

Updated PNNI physical link. Traffic metric parameters are sent to the layer manager. These parameters are required ICC supports PNNI routing.

```
typedef struct pnMtrcCfg
{
   U8               qos;       /* quality of service */
   U8               clp;       /* cell loss priority, 0/1 */
   PnTfcMtrc        mtrc;      /* traffic metrics */
} PnMtrcCfg;
```

qos

QoS service type for which these traffic metrics apply. The allowable values are:

Value	Description
PN_QOS_CLASS_UBR	Unspecified
PN_QOS_CLASS_CBR	Constant bit rate
PN_QOS_CLASS_VBR_RT	Variable bit rate—real time
PN_QOS_CLASS_VBR_NRT	Variable bit rate—**n** real time
PN_QOS_CLASS_ABR	Available bit rate

clp

GCAC cell loss priority bit. The allowable value: 0 or 1.

mtrc

```
typedef struct pnTfcMtrc
{
   U32   adminWt;    /* administrative weight */
   U32   maxCR;      /* max cell rate (cells/sec) */
   U32   avCR;       /* available cell rate (cells/sec) */
   U32   ctd;        /* cell transfer delay (10 u-sec) */
   U32   cdv;        /* cell delay variation (10 u-sec) */
   U16   clr0;       /* cell loss ratio CLP = 0 */
   U16   clr1;       /* cell loss ratio CLP = 0+1 */
   U32   crm;        /* opt GCAC : cell rate margin */
   U32    vf;        /* opt GCAC : variance factor (2**-8) */
   U16   tmFlgs;     /* GCAC parameters present */
} PnTfcMtrc;
```

adminWt

Administrative weight.

maxCR

Maximum cell rate supported by this link, in cells/sec.

avCR

Available cell rate, in cells/sec.

ctd

Cell transmission delay, units of 10 microseconds.

cdv

Cell delay variation, units of 10 microseconds.

clr0

Cell loss ratio, when cell loss priority is 0.

clr1

Cell loss ratio, when cell loss priority is 0+1.

crm

Cell rate margin: Optional GCAC parameter.

vf

Variance factor optional GCAC parameter.

tmFlgs

Optional generic call admission control parameters are present. The allowable values are:

Value	Description
PN_TM_FLG_GCAC_PRES	Present
NOTPRSNT	Not present

Note: *The remaining fields are the same as those passed in the configuration request.*

Description:

The RM uses this primitive to indicate to the layer manager the result of a configuration request.

Returns:

00 ROK

01 RFAILED

3.7.1.2.3 RmMiLrmCntrlReq

Name:

Control Request

Direction:

Layer manager to the RM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RmMiLrmCntrlReq(pst, cntrl)
Pst     *pst;
RtMngmt *cntrl;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. The control structure has the following format:

```
typedef struct rmMngmt
{
   Header    hdr;                 /* header */
   CmStatus  cfm;                 /* status in confirm */
   union
   {
 /* Control */
      struct
      {
         DateTime dt;             /* date and Time */
         U8 action;               /* Action */
         U8 subAction;            /* SubAction */
         union
         {
            RmBbPhyCntrl rmBbPhyCntrl; /* BB Physical Link Control */
            RmBbIntfcCntrl rmBbIntfcCntrl; /* BB Interface Control */
            RmNbDpcCntrl rmNbDpcCntrl; /* NB DPC Control */
            RmVpCntrl    rmVpCntrl;    /* VPCI Control */
            RmVcCntrl    rmVcCntrl;    /* VCI COntrol */
            RmCicCntrl   rmCicCntrl;   /* CIC Control */
            RmPvcCntrl   rmPvcCntrl;   /* PVC Control */
            RmDss1IntfcCntrl rmDss1IntfcCntrl;
                                       /* DSS1 Interface Control */
            U32          dbgMask;      /* debug mask */

            /* Following field is significant in case of CntrlCfm */
            RmDiag       diagn;        /* Diagnostics */
            RmObsTrc     rmObsTrc;     /* Observation Index Tracing */
            RmUpSapCntrl rmUpSAPCntrl; /* Up SAP Control */
#ifdef ICC_AUDIT
            RmAuditCntrl   rmAuditCntrl;   /* Audit Resource */
#endif /* ICC_AUDIT */
            RmGrpSapCntrl rmGrpSapCntrl;   /* group sap control */
         }s;
      } cntrl;
   }t;
} RmMngmt;
```

hdr

The **elmnt** field in the element ID (**elmId**) structure defines the element. For more description, see Section 3.3.1, "Header."

The allowable values for this primitive are:

Value	Description
STRMBBPHY	Broadband physical link
STRMNBDPC	Narrowband DPC
STRMVP	VPCI
STRMCIC	CIC
STRMPVC	PVC (static binding)
STRMVC	VC
STRMDSS1INTFC	DSS1 interface
STRMBBINTFC	Broadband interface

cfm

It is not relevant to this request.

dt

Date and time structure.

action

Specific action code. The allowable values are:

Name	Description
AENA	Enable
ADISIMM	Disable immediately
ADEL	Delete
ARST	Reset
AADD	Add
STRMAMOD	Modify the element
AUBND_DIS	Unbind disable
ASHUTDOWN	Shutdown
AENAINTFC	Enable the interface
ADISAINTFC	Disable the interface

subAction

The allowable values are:

Name	Description
SAUSTA	Unsolicited status generation
SAELMNT	Specific element
SADBG	Debug option
SATRC	Trace control
SAAUD	Audit control
SAGR_DSTPROCID	Group control based on destination process ID

rmBbPhyCntrl

This is the information required to control a physical broadband link. A physical broadband link can be deleted via the control request.

```
typedef struct rmBbPhyCntrl
{
   U32   intfcId;               /* Physical Interface Identifier */
} RmBbPhyCntrl;
```

intfcId

Specifies the physical broadband link. This value must be the same as that passed in the configuration request. For more information, refer to Section 3.7.1.2.1, "**RmMiLrmCfgReq**."

rmBbIntfcCntrl

Information required to control a broadband DPC. A broadband DPC can be deleted via the control request.

```
typedef struct rmBbIntfcCntrl
{
   RmInterface bbIntfc;           /* Broadband Interface */
} RmBbIntfcCntrl;
```

bbIntfc

Broadband interface. It is defined as:

```
typedef struct rmInterface    /* Generic Interface Structure */
```

```
{
   U8  intfType;              /* Identifies the Interface type */
   union interface
   {
     Dpc   dpc;               /* For ISUP, BISUP Interface type */
     U32   intfcId;           /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The allowable values of the interface type are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP interface.

intfcId

Interface ID. The allowable value: 32-bit integer.

rmNbDpcCntrl

Information required to control a narrowband DPC. A narrowband DPC can be deleted via the control request.

```
typedef struct rmNbDpcCntrl     /* Narrowband DPC Control Structure */
{
   Dpc dpc;                     /* DPC */
   U32 cotFrequency;            /* Continuity check allowed */
   U8  cotChkFlag;              /* Continuity check allowed */
   U8  echoCntrlFlag;           /* Echo control allowed */
} RmNbDpcCntrl;
```

dpc

Specifies the narrowband DPC.

cotFrequency

For more information, refer to the description of the **cotFrequency** on page 191.

cotChkFlag

For more information, refer to the description of the **cotChkFlag** on page 195.

echoCntrlFlag

For more information, refer to the description of the **echoCntrlFlag** on page 214.

rmVpCntrl

Information required to control a broadband VPI. A broadband VPI can be deleted, enabled, or disabled.

```
typedef struct rmVpCntrl
{
   RmInterface bbIntfc;           /* Broadband Interface */
   VpId   vpId;                   /* VPCI */
} RmVpCntrl;
```

bbIntfc

Broadband interface to which this VPI belongs.

```
typedef struct rmInterface   /* Generic Interface Structure */
{
   U8  intfType;             /* Identifies the Interface type */
   union interface
   {
     Dpc   dpc;              /* For ISUP, BISUP Interface type */
     U32   intfcId;          /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The allowable values for this interface type are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP interface.

intfcId

Interface ID. The allowable value: 32-bit integer.

vpId

VPI ID.

rmVcCntrl

Information required to reset a broadband bearer channel VPI/VCI. To reset a VPI/VCI means to make it available immediately in the RM for further connections. The VPI/VCI are marked as idle and available. Messages are not sent to GCC or the protocol entities. The layer manager must make sure that the call is cleared on this channel and that there is no inconsistency between the different entities.

```
typedef struct rmVcCntrl
{
   RmInterface bbIntfc;           /* Broadband Interface */
   VpId   vpId;                   /* VPCI */
   VcId   vcId;                   /* VCC Id */
} RmVcCntrl;
```

bbIntfc

Broadband interface to which this VPI/VCI belongs.

```
typedef struct rmInterface   /* Generic Interface Structure */
{
   U8  intfType;             /* Identifies the Interface type */
   union interface
   {
     Dpc   dpc;              /* For ISUP, BISUP Interface type */
     U32   intfcId;          /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The allowable values for this interface type are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP interface.

intfcId

Interface ID. The allowable value: 32-bit integer.

vpId

VPI ID.

vcId

VCI ID.

rmCicCntrl

Information required to control one or more narrowband circuit(s). A circuit can be deleted, enabled, disabled, or reset via the control request. A circuit can be deleted only if it is idle, and as long as the circuit is busy, it cannot be deleted.

To *disable* a circuit means that the circuit is not available for subsequent allocation.

To *enable* a circuit means to make it available for subsequent allocation. If a call is associated with the specified circuit, the circuit is not available for allocation until the call is cleared.

To *reset* a circuit means to make it available immediately in the RM for further connections. The circuit is marked as idle and available. Messages are not sent to GCC or the protocol entities. The layer manager must make sure that the call is cleared on this circuit and that there is no inconsistency between the different entities.

```
typedef struct rmCicCntrl
{
   Dpc    dpc;                  /* DPC */
   U16    cic;                  /* Starting CIC */
   U16    numCic;               /* Number of CIC */
   Ticks  delayVal;             /* Delay over this circuit */
   U8     cotChkFlag;           /* Continuity check allowed */
   U8     echoCntrlFlag;        /* Echo control allowed */
} RmCicCntrl;
```

dpc

DPC to which this VPI belongs.

cic

CIC value. The control request affects all the circuits beginning with this start CIC.

numCic

Number of circuits affected by the control request. The number of circuits includes the start circuit.

delayVal

For more information, refer to the description of the **delayVal** on page 195.

cotChkFlag

For more information, refer to the description of the **cotChkFlag** on page 195.

echoCntrlFlag

For more information, refer to the description of the **echoCntrlFlag** on page 214.

rmPvcCntrl

Information required to delete a static binding.

```
typedef struct rmPvcCntrl
{
   RmRsc  rsrc;
} RmPvcCntrl;
```

rsrc

Resource for which static binding is deleted. This can be any pair of resources that are bound together. For further details on specifying the resource, refer to Section 3.7.1.2.1, "**RmMiLrmCfgReq**."

rmNbDss1IntfcCntrl

Information required to control a DSS1 interface. A DSS1 interface can be deleted via the control request. Specified channels of the DSS1 interface can be disabled, enabled, reset, or marked as equipped or unequipped.

```
typedef struct rmDss1IntfcCntrl
{
   U32 intfcId;             /* DSS1 Interface Id */
   U8  intId;              /* Interface Id as defined for NFAS Access */
   U8 chnl[LRMMAXPRICHNL];  /* Array of channels defined on the DSS1 */
} RmDss1IntfcCntrl;
```

intfcId

The DSS1 interface ID.

intId

The interface ID for the PRI link at the DSS1 interface. This parameter is valid if a multiple PRI access is used for the DSS1 interface.

chnl

List identifying the channels of the given DSS1 interface to be enabled, disabled, reset, or marked as equipped / unequipped, via this control request. **chnl[i]** is set to **TRUE** if the control request procedure applies to channel **i**; otherwise, it is set to **FALSE**.

dbgMask

This field is reserved for future releases and is currently not used.

diagn

This field is significant only to the control confirm primitive. For more description, see Section 3.7.1.2.4, "**RmMiLrmCntrlCfm**."

rmObsTrc

This control is for setting a trigger, based on the particular resource information.

```
typedef struct rmObsTrc
{
   U8 objType;       /* Type of object to be observed */
   RmRsc rsc;       /* resource */
   U8 obsIdx;        /* resource observation index */
} RmObsTrc;
```

objType

Type of object observed for signalling conversion analysis. The allowable values are:

LRM_INTFC
LRM_BB_VPI
LRM_RSC

rsc

The resource information on which the trigger must be set.

obsIdx

For the incoming resource or an interface, this index determines the row in the observation trigger table within GCC that is associated with this incoming resource. If an observation trigger is not required for this resource, then the value of **obsIdx** should be 0.

For the outgoing resource or an interface, this index determines the column in the observation trigger table within GCC that is associated with this outgoing resource. If an observation trigger is not required for this resource, then the value of **obsIdx** should be 0.

rmUpSAPCntrl

SAP control request.

```
typedef struct rmUpSAPCntrl
{
   SpId sapId;
} RmUpSAPCntrl;
```

sapId

SAP ID of the SAP on which this control request applies.

rmAuditCntrl

Control request for auditing.

```
typedef struct rmAuditCntrl
{
   SpId sapId;                      /* SAP Id for this control */
   RmAuditRscGrp rmAuditRscGrp;  /* Reosurce Group */
} RmAuditCntrl;
```

sapId

The ID of the SAP for which auditing is requested.

rmAuditRscGrp

The resource group for this auditing. It is used only in the OAP. Refer to the *Interworking Call Control Interface Service Definition* for more details.

rtGrpSapCntrl

Group SAP control request.

```
typedef struct rmGrpSapCntrl
{
   ProcId dstProcId;
} RmGrpSapCntrl;
```

dstProcId

The destination process ID of the entity with which the group of SAPs are bound.

Description:

This function is used to control the RM. The possible operations with required parameters are listed in the following table.

Description	**subAction**	**action**	**elmnt**	**Others**
Enable alarms	**SAUSTA**	**AENA**	N / A	N / A
Disable alarms		**ADISIMM**		
Enable a debug class	**SADBG**	**AENA**		**dbgMask**
Disable a debug class		**ADISIMM**		
Delete a physical broadband link	**SAELMNT**	**ADEL**	**STRMBBPHY**	**rmBbPhyCntrl**
Delete a broadband **Intfc**		**ADEL**	**STRMBBINTFC**	**rmBbIntfcCntrl**
Delete a narrowband DPC		**ADEL**	**STRMNBDPC**	**rmNbDpcCntrl**
Delete a VPI		**ADEL**	**STRMVP**	**rmVpCntrl**
Make a VPI available for allocation		**AENA**		
Make a VPI unavailable for allocation		**ADISIMM**		
Free a VCI. Remove the allocated status.		**ARST**	**STRMVC**	**rmVcCntrl**
Delete a circuit		**ADEL**	**STRMCIC**	**rmCicCntrl**
Make a circuit available for allocation		**AENA**		
Make a circuit unavailable for allocation		**ADISIMM**		
Free a circuit. Remove the allocated status.		**ARST**		
Delete a static binding		**ADEL**	**STRMPVC**	**rmPvcCntrl**
Delete a DSS1 interface		**ADEL**	**STRMDSS1INTFC**	**rmDss1IntfcCntrl**
Disable the specified channels of a DSS1 interface		**ADISIMM**	**STRMDSS1INTFC**	**rmDss1IntfcCntrl**
Enable the specified channels of a DSS1 interface		**AENA**	**STRMDSS1INTFC**	**rmDss1IntfcCntrl**

Description	subAction	action	elmnt	Others
Reset the specified channels of a DSS1 interface		**ARST**	**STRMDSS1INTFC**	**rmDss1IntfcCntrl**
Equip the specified channels of a DSS1 interface		**AADD**	**STRMDSS1INTFC**	**rmDss1IntfcCntrl**
Modify the configuration	**SAELMNT**	**STRMAMOD**	**STRMNBDPC, STRMCIC**	**rmNbDpcCntrl, rmCicCntrl**
Delete an UP SAP	**SAELMNT**	**ADEL**	**STRMUPSAP**	**rmUpSAPCntrl**
Unbind disable UP SAP	**SAELMNT**	**AUBND_DIS**	**STRMUPSAP**	**rmUpSAPCntrl**
Shut down the RM entity	**SAELMNT**	**ASHUTDOWN**	**STGEN**	**N/A**
Enable the PAP audit	**SAAUD**	**AENA**	**STRMPAPAUD**	**rmAuditCntrl**
Enable the GAP audit	**SAAUD**	**AENA**	**STRMGAPAUD**	**rmAuditCntrl**
Enable the OAP audit	**SAAUD**	**AENA**	**STRMOAPAUD**	**rmAuditCntrl**
Disable the PAP audit	**SAAUD**	**ADISIMM**	**STRMPAPAUD**	**rmAuditCntrl**
Disable the GAP audit	**SAAUD**	**ADISIMM**	**STRMGAPAUD**	**rmAuditCntrl**
Disable the OAP audit	**SAAUD**	**ADISIMM**	**STRMOAPAUD**	**rmAuditCntrl**
Group SAP unbinding disable	**SAGR_DSTPROCID**	**AUBND_DIS**	**STGRRMSAP**	**rmGrpSapCntrl**

Returns:

00 **ROK**

01 **RFAILED**

3.7.1.2.4 RmMiLrmCntrlCfm

Name:

Control Confirm

Direction:

RM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RmMiLrmCntrlCfm(pst, cntrl)
Pst     *pst;
RtMngmt *cntrl;
```

Parameters:

pst

For more description, refer to Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. With the exception of the fields described next, the structure used for the control confirm is the same as that for the control request. For more information, see Section 3.7.1.2.3, "**RmMiLrmCntrlReq**."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
    U16   status;              /* Status of the operation  */
    U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible:

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_INVALID_ACTION	Invalid action passed in the control structure
LCM_REASON_INVALID_SUBACTION	Invalid subaction passed in the control structure
LCM_REASON_GENCFG_NOT_DONE	General configuration must be done before a control request can be processed
LCM_REASON_INVALID_PAR_VAL	One of the passed parameters is invalid. The **diagn** field has more specific information about the parameter that caused the failure.
LCM_REASON_LRM_EPHYBSY	Physical interface cannot be deleted—one or more VPIs are assigned to it.
LCM_REASON_LRM_EDPCBSY	The DPC cannot be deleted—one or more resources are assigned to it.
LCM_REASON_LRM_ERSCBSY	Resource cannot be deleted—the specified resource is involved in a call.

Name	Description
LCM_REASON_LRM_EPART_SUCC	A specified circuit could not be deleted because it is busy.
LCM_REASON_INVALID_ELMNT	Element is invalid
LCM_REASON_LRM_ECHNLBSY	DSS1 interface cannot be deleted—one or more DSS1 channels of the DSS1 interface is busy with a call.
LCM_REASON_LRM_AUD_REPEAT_REQ	Unfinished, exact type auditing procedure on the same SAP

diagn

Provides further information about the error. This field is significant only when the status indicates that the request failed. See **diagn** on page 203 for a description.

Description:

The RM uses this primitive to indicate to the layer manager the result of a control request.

Returns:

00 **ROK**

01 **RFAILED**

3.7.1.2.5 RmMiLrmStsReq

Name:

Statistics Request

Direction:

Layer manager to the RM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RmMiLrmStsReq(pst, action, sts)
Pst      *pst;
Action   action;
RtMngmt *sts;
```

Parameters:

pst

For more description, refer to Section 3.3.3, "**Pst**."

action

Action indicator. The allowable values are:

Name	Description
0	Zero statistics counters (**ZEROSTS**)
1	Do not set the statistics counters to zero (**NOZEROSTS**)

sts

Pointer to the statistics structure. The statistics structure has the following format:

```
typedef struct rmMngmt
{
   Header hdr;
   CmStatus  cfm;                 /* status in confirm */
   union
   {
/* Statistics */
      struct
      {
         DateTime dt;            /* Date and Time */
         union
         {
            RmNbDpcSts  rmNbDpcSts;
            RmBbIntfcSts rmBbIntfcSts;
         } s;
      }sts;
   }t;
} RmMngmt;
```

hdr

The **elmnt** field in the element ID (**elmId**) structure defines the element. For more description, see Section 3.3.1, "Header."

The allowable values are:

Value	Description
STRMBBINTFC	Broadband INTFC
STRMNBDPC	Narrowband DPC

cfm

Valid only in confirm primitives.

dt

Date and time structure.

rmNbDpcSts

Narrowband DPC statistics. For the statistics request, only the **dpc** field is significant. The RM sets the other fields, which are also passed to the layer manager in the statistics confirm request. For more details, see Section 3.7.1.2.6, "**RmMiLrmStsCfm**."

```
typedef struct rmNbDpcSts
{
   Dpc dpc;
   U32 alocReq;
   U32 alocSucc;
} RmNbDpcSts;
```

dpc

DPC for which statistics are requested.

alocReq, alocSucc

It is not used in the request.

rmBbIntfcSts

Broadband DPC statistics. For the statistics request, only the **dpc** field is significant. The other fields are set by the RM and passed to the layer manager in the statistics confirm request. For more details, see Section 3.7.1.2.6, "**RmMiLrmStsCfm**."

```
typedef struct rmBbIntfcSts
{
   RmInterface bbIntfc;
   U32 alocReq;
   U32 alocSucc;
} RmBbIntfcSts;
```

bbIntfc

Broadband interface for which the statistics are requested.

```
typedef struct rmInterface   /* Generic Interface Structure */
{
   U8  intfType;             /* Identifies the Interface type */
   union interface
   {
     Dpc   dpc;              /* For ISUP, BISUP Interface type */
     U32   intfcId;          /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The interface type. The allowable values are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP interface.

intfcId

Interface ID. The allowable value: 32-bit integer.

alocReq, alocSucc

It is not used in the request.

Description:

The layer manager uses this function to gather statistics information about a particular interface.

Returns:

00 ROK

01 RFAILED

3.7.1.2.6 RmMiLrmStsCfm

Name:

Statistics Confirm

Direction:

RM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RmMiLrmStsCfm(pst, sts)
Pst     *pst;
RmMngmt *sts;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

sts

Pointer to the statistics structure. The statistics structure has the following format:

```
typedef struct rmMngmt
{
   Header hdr;
   CmStatus  cfm;                 /* status in confirm */
   union
   {
/* Statistics */
      struct
      {
         DateTime dt;            /* Date and Time */
         union
         {
            RmNbDpcSts   rmNbDpcSts;
            RmBbIntfcSts rmBbIntfcSts;
         } s;
      }sts;
    } t;
} RmMngmt;
```

hdr

For a description, refer to Section 3.3.1, "Header."

cfm

Status field. It indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
   U16   status;              /* Status of the operation  */
   U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Statistics request successful
LCM_PRIM_NOK	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_GENCFG_NOT_DONE	General configuration must be done before a control request can be processed
LCM_REASON_INVALID_PAR_VAL	A passed parameter is invalid

dt

Date and time structure.

rmNbDpcSts

Narrowband DPC statistics.

```
typedef struct rmNbDpcSts
{
   Dpc dpc;
   U32 alocReq;
   U32 alocSucc;
} RmNbDpcSts;
```

dpc

DPC for which the statistics are requested.

alocReq

Number of allocation requests for resources toward this DPC.

alocSucc

Number of successful resource allocations toward this DPC.

rmBbIntfcSts

Broadband interface statistics.

```
typedef struct rmBbIntfcSts
{
   RmInterface bbIntfc;
   U32 alocReq;
   U32 alocSucc;
} RmBbIntfcSts;
```

bbIntfc

Interface for which statistics are requested.

alocReq

Number of allocation requests for resources toward this interface.

alocSucc

Number of successful resource allocations toward this interface.

Description:

The RM uses this function to provide the layer manager with statistics information at a particular interface.

Returns:

00 **ROK**

01 **RFAILED**

3.7.1.2.7 RmMiLrmStaReq

Name:

Status Request

Direction:

Layer manager to the RM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 RmMiLrmStaReq(pst, sta)
Pst      *pst;
RmMngmt  *sta;
```

Parameters:

pst

For more description see Section 3.3.3, "**Pst**."

sta

Pointer to the solicited status structure. It has the following format.

```
typedef struct rmMngmt
{
   Header hdr;
   CmStatus  cfm;                  /* status in confirm */
   union
   {
/* Solicited Status */
      struct
      {
         DateTime dt;              /* Date and Time */
         union
         {
            RmVpSta  rmVpSta;      /* VPCI Status */
            RmVcSta  rmVcSta;      /* VCI Status */
            RmCicSta rmCicSta;     /* CIC Status */
            RmDss1IntfcSta rmDss1IntfcSta;
                                   /* Status of the DSS1 Interface */
            RmChnlSta rmChnlSta;   /* chnl status */
         }s;
      } ssta;
   } t;
} RmMngmt;
```

hdr

Header structure. For more description, see Section 3.3.1, "Header." The **elmnt** field in the element ID (**elmId**) structure defines the element. The allowable values are:

Value	Description
STRMUPSAP	Call control SAP element
STRMBBPHY	Broadband physical link
STRMNBDPC	Narrowband DPC
STRMVP	VPCI
STRMCIC	CIC
STRMPVC	PVC (static binding)
STRMVC	VC
STRMDSS1INTFC	DSS1 interface
STRMBBINTFC	Broadband interface
STRMOBS	Observation trigger index
STRMAUDPAP	Periodic auditing
STRMAUDOAP	One-time auditing
STRMAUDGAP	GCC auditing
STGRRMSAP	Group RM SAP
STRMUPSAP	RM SAP
STRMCHNL	DSS1 channel

cfm

It is not valid in the status request.

dt

Date and time structure.

rmVpSta

VPI status structure. The **bbIntfc** and **vpId** fields specify the VPI for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. See Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct rmVpSta
{
   RmInterface bbIntfc;          /* Broadband Interface */
   U16    vpId;                  /* VPCI */
   U8     state;                 /* State of VPCI */
   U8     actvCalls;             /* Calls Active on this VP */
   U32    cfgFCR[MAX_QOS_CLASSES];    /* Configured FCR */
   U32    cfgBCR[MAX_QOS_CLASSES];    /* Configured BCR */
   U32    usdFCR[MAX_QOS_CLASSES];    /* Used Forward Cell Rate */
   U32    usdBCR[MAX_QOS_CLASSES];    /* Used Backward Cell Rate */
   U8     obsIdx;                 /* resource observation index */
} RmVpSta;
```

rmVcSta

Broadband channel (VPI/VCI) status structure. The **bbIntfc**, **vpId**, and **vcId** fields specify the channel for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. See Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct rmVcSta
{
   RmInterface bbIntfc;           /* Broadband Interface */
   VpId   vpId;                   /* VPCI */
   VcId   vcId;                   /* VCC Id */
   U8     state;                  /* State */
   UConnId suConnId;              /* User Holding the Resource */
   U8     qos;                    /* quality of service */
   U8 obsIdx;                     /* resource observation index */
} RmVcSta;
```

rmCicSta

Circuit status structure. The **dpc** and **cic** fields specify the circuit for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. See Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct rmCicSta
{
   Dpc    dpc;                  /* DPC */
   U16    cic;                  /* Circuit Identification Code */
   U8     state;                /* State of CIC */
   UConnId suConnId;            /* User Holding the Resource */
   U8 obsIdx;                   /* resource observation index */
   U8   mgId;                   /* Media Gateway Id */
} RmCicSta;
```

rmDss1IntfcSta

DSS1 interface status structure. The **intfcId**, **intId**, and **accessType** fields specify the DSS1 interface for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. See Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct rmDss1IntfcSta
{
   U32 intfcId;            /* Interface Id for which status requested */
   U8 intId;              /* Interface Identifier for NFAS Access */
   U8 accessType;         /* Access Type */
   U8 alocMeth;           /* Channel Allocation Method used */
   struct chnlSta
   {
      U8 state;                 /* State of Channel */
      UConnId suConnId;         /* User Holding the Channel */
      U8 pvc;                   /* If a PVC is assocaited */
      U8 obsIdx;                /* resource observation index */
   } chnl[LRMMAXPRICHNL];
   U8 obsIdx;                   /* resource observation index */
   U8   mgId;                   /* Media Gateway Id */
} RmDss1IntfcSta;
```

rmChnlSta

Dss1 channel status structure. The **intfcId** and **chnlId** fields specify the channel whose status is requested. The **intId** is used only for the NFAS interface, with the **intfcId** and **chnlId** to identify the channel. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. See Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct rmChnlSta
{
   U32 intfcId;            /* Interface Id for which status requested */
   U8 intId;              /* Interface Identifier for NFAS Access */
   U8 chnlId;
   struct
   {
      U8 state;                 /* State of Channel */
      UConnId suConnId;         /* User Holding the Channel */
      U8 pvc;                   /* If a PVC is assocaited */
      U8 obsIdx;                /* resource observation index */
   } chnl;
} RmChnlSta;
```

Description:

The layer manager uses this function to gather solicited status information.

Returns:

00 **ROK**

01 **RFAILED**

3.7.1.2.8 RmMiLrmStaCfm

Name:

Status Confirm

Direction:

RM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 RmMiLrmStaCfm(pst, sta)
Pst     *pst;
RmMngmt *sta;
```

Parameters:

pst

For more details, see Section 3.3.3, "**Pst**."

sta

Pointer to the solicited status structure. The solicited status structure has the following format:

```
typedef struct rmMngmt
{
   Header hdr;
   CmStatus  cfm;                  /* status in confirm */
   union
   {
/* Solicited Status */
      struct
      {
         DateTime dt;              /* Date and Time */
         union
         {
            RmVpSta  rmVpSta;      /* VPCI Status */
            RmVcSta  rmVcSta;      /* VCI Status */
            RmCicSta rmCicSta;     /* CIC Status */
            RmDss1IntfcSta rmDss1IntfcSta;
                                   /* Status of the DSS1 Interface */
            RmChnlSta rmChnlSta;   /* chnl status */
         }s;
      } ssta;
   } t;
} RmMngmt;
```

hdr

Header structure. For more information, refer to Section 3.3.1, "Header."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
    U16   status;              /* Status of the operation  */
    U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Statistics request successful
LCM_PRIM_NOK	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible:

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_GENCFG_NOT_DONE	General configuration must be done before a control request can be processed
LCM_REASON_INVALID_PAR_VAL	Passed parameters are invalid. The specified resource is not configured.

dt

Date and time structure.

rmVpSta

VPI status.

```
typedef struct rmVpSta
{
   RmInterface bbIntfc;            /* Broadband Interface */
   U16    vpId;                    /* VPCI */
   U8     state;                   /* State of VPCI */
   U8     actvCalls;               /* Calls Active on this VP */
   U32    cfgFCR[MAX_QOS_CLASSES]; /* Configured FCR */
   U32    cfgBCR[MAX_QOS_CLASSES]; /* Configured BCR */
   U32    usdFCR[MAX_QOS_CLASSES]; /* Used Forward Cell Rate */
   U32    usdBCR[MAX_QOS_CLASSES]; /* Used Backward Cell Rate */
   U8     obsIdx;                  /* resource observation index */
} RmVpSta;
```

bbIntfc

Broadband interface to which this VPI belongs.

```
typedef struct rmInterface    /* Generic Interface Structure */
{
   U8  intfType;        /* Identifies the Interface type */
   union interface
   {
      Dpc   dpc;              /* For ISUP, BISUP Interface type */
     U32   intfcId;           /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The interface type. The allowable values are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP interface.

intfcId

Interface ID. The allowable value: 32 bit integer.

vpId

The Virtual Path ID (VPI).

state

The state of the VPI. The state can either be available (**TRUE**) or not available (**FALSE**).

actvCalls

Indicates whether there are calls on this VPI. **TRUE** means that there are active calls on this VPI and **FALSE** means that the VPI is idle.

cfgFCR

Configured maximum forward cell rate. The maximum bandwidth is maintained per QoS service class. The following service classes are possible.

Value	Description
PN_QOS_CLASS_UBR	Unspecified
PN_QOS_CLASS_CBR	Constant bit rate
PN_QOS_CLASS_VBR_RT	Variable bit rate—real time
PN_QOS_CLASS_VBR_NRT	Variable bit rate—**n** real time
PN_QOS_CLASS_ABR	Available bit rate

cfgBCR

Configured maximum backward cell rate. The maximum bandwidth is maintained per QoS service class. The following service classes are possible.

Value	Description
PN_QOS_CLASS_UBR	Unspecified
PN_QOS_CLASS_CBR	Constant bit rate
PN_QOS_CLASS_VBR_RT	Variable bit rate—real time
PN_QOS_CLASS_VBR_NRT	Variable bit rate—**n** real time
PN_QOS_CLASS_ABR	Available bit rate

usdFCR

Used forward cell rate. The cell rate is maintained per QoS service class. The following service classes are possible.

Value	Description
PN_QOS_CLASS_UBR	Unspecified
PN_QOS_CLASS_CBR	Constant bit rate
PN_QOS_CLASS_VBR_RT	Variable bit rate—real time
PN_QOS_CLASS_VBR_NRT	Variable bit rate—**n** real time
PN_QOS_CLASS_ABR	Available bit rate

usdBCR

Used backward cell rate. The cell rate is maintained per QoS service class. The following service classes are possible.

Value	Description
PN_QOS_CLASS_UBR	Unspecified
PN_QOS_CLASS_CBR	Constant bit rate
PN_QOS_CLASS_VBR_RT	Variable bit rate—real time
PN_QOS_CLASS_VBR_NRT	Variable bit rate—**n** real time
PN_QOS_CLASS_ABR	Available bit rate

obsIdx

Refer to **rmObsTrc** on page 216 for further details.

rmVcSta

Broadband channel (VPI/VCI) status.

```
typedef struct rmVcSta
{
   RmInterface bbIntfc;          /* Broadband Interface */
   VpId    vpId;                 /* VPCI */
   VcId    vcId;                 /* VCC Id */
   U8      state;                /* State */
   U8     qos;                   /*   quality of service */
   UConnId suConnId;             /* User Holding the Resource */
   U8      obsIdx;               /* resource observation index */
} RmVcSta;
```

bbIntfc

Broadband interface to which this VPI/VCI belongs.

```
typedef struct rmInterface    /* Generic Interface Structure */
{
   U8  intfType;        /* Identifies the Interface type */
   union interface
   {
     Dpc   dpc;              /* For ISUP, BISUP Interface type */
     U32   intfcId;           /* DSS1, DSS2  Interfaces */
   }t;
} RmInterface;
```

intfType

The interface type. The allowable values are:

CC_BI_INTFC
CC_AM_INTFC

dpc

Destination point code of the B-ISUP interface.

intfcId

Interface ID. The allowable value: 32-bit integer.

vpId

The virtual path ID (VPI).

vcId

The virtual channel ID (VCI).

state

State of the VPI/VCI. The state can have one of the following values:

Value	Description
LRM_IDLE	Resource is IDLE
LRM_UNEQUIP	Resource is unequipped
LRM_CP_BSY	Resource is CP busy
LRM_MNT_BSY	Resource is maintenance busy

suConnId

Connection ID of the connection to which the VCC is currently allocated.

qos

qos of the connection to which the VCC is currently allocated.

obsIdx

Refer to **rmObsTrc** on page 216 for more details.

rmCicSta

Circuit status.

```
typedef struct rmCicSta
{
   Dpc     dpc;                    /* DPC */
   U16     cic;                    /* Circuit Identification Code */
   U8      state;                  /* State of CIC */
   UConnId suConnId;               /* User Holding the Resource */
   U8      obsIdx;                 /* resource observation index */
   U8      mgId;                   /* Media Gateway Id */
} RmCicSta;
```

dpc

DPC to which this circuit belongs.

cic

Circuit ID.

state

The state of the circuit. The state can have one of the following values:

Value	Description
LRM_IDLE	Resource is IDLE
LRM_UNEQUIP	Resource is unequipped
LRM_CP_BSY	Resource is CP busy
LRM_MNT_BSY	Resource is maintenance busy

suConnId

Connection ID of the connection to which the circuit is currently allocated.

obsIdx

Refer to **rmObsTrc** on page 216 for more details.

mgId

Media gateway ID. The RM uses this while allocating resources for a call from the same media gateway.

rmDss1IntfcSta

DSS1 interface status.

```
typedef struct rmDss1IntfcSta
{
   U32 intfcId;              /* Interface Id for which status requested
                              */
   U8 intId;                 /* Interface Identifier for NFAS Access */
   U8 accessType;            /* Access Type */
   U8 alocMeth;              /* Channel Allocation Method used */
   struct chnlSta
   {
      U8 state;              /* State of Channel */
      UConnId suConnId;      /* User Holding the Channel */
      U8 pvc;                /* If a PVC is associated */
      U8 obsIdx;             /* resource observation index */
   } chnl[LRMMAXPRICHNL];
   U8 obsIdx;                /* resource observation index */
   U8   mgId;                /* Media Gateway Id */
} RmDss1IntfcSta;
```

intfcId

DSS1 interface ID.

intId

Interface ID for the PRI link in the DSS1 interface. This parameter is valid if a multiple-PRI access is used for the DSS1 interface.

accessType

Indicates the type of access used—PRI, BRI, or multiple-PRI.

alocMeth

Resource allocation method used for this interface.

chnl

Information regarding the DSS1 channels of this interface. The following information is available.

state

State of the DSS1 channel. The state can have one of the following values:

Value	Description
LRM_IDLE	Resource is IDLE
LRM_UNEQUIP	Resource is unequipped
LRM_CP_BSY	Resource is CP busy
LRM_MNT_BSY	Resource is maintenance busy

suConnId

Connection ID of the connection to which this channel is currently allocated.

pvc

Pointer to the PVC control block associated with the DSS1 channel.

obsIdx

Refer to **rmObsTrc** on page 216 for more information.

mgId

Media gateway ID. The RM uses this while allocating resources for a call from the same media gateway.

rmChnlSta

```
typedef struct rmChnlSta
{
   U32 intfcId;            /* Interface Id for which status requested */
   U8 intId;               /* Interface Identifier for NFAS Access */
   U8 chnlId;
   struct
   {
      U8 state;                  /* State of Channel */
      UConnId suConnId;          /* User Holding the Channel */
      U8 pvc;                    /* If a PVC is assocaited */
      U8 obsIdx;                 /* resource observation index */
   } chnl;
} RmChnlSta;
```

intfcId

DSS1 interface ID.

intId

Interface ID of the PRI link at the DSS1 interface. This parameter is valid if a multiple-PRI access is used for the DSS1 interface.

chnl

Information regarding the DSS1 channel. The following information is available.

state

State of the DSS1 channel. The state can have one of the following values:

Value	Description
LRM_IDLE	Resource is IDLE
LRM_UNEQUIP	Resource is unequipped
LRM_CP_BSY	Resource is CP busy
LRM_MNT_BSY	Resource is maintenance busy

suConnId

Connection ID of the connection to which this channel is currently allocated.

pvc

Pointer to the PVC control block associated with the DSS1 channel.

obsIdx

Refer to **rmObsTrc** on page 216 for more information.

Description:

The RM uses this function to return solicited status information to the layer manager.

Returns:

00 **ROK**

01 **RFAILED**

3.7.1.3 RmMiLrmStaInd

Name:

Status Indication

Direction:

RM to the layer manager

Supplied:

In the layer manager

Synopsis:

```
PUBLIC S16 RmMiLrmStaInd(pst, sta)
Pst     *pst;
RmMngmt *sta;
```

Parameters:

pst

For more information, see Section 3.3.3, "**Pst**."

sta

Pointer to the status structure. Status structure has the following format.

```
typedef struct rmMngmt
{
   Header hdr;
   CmStatus  cfm;                 /* status in confirm */
   union
   {
/* UnSolicited Status */
      struct
      {
         CmAlarm alarm;            /* Alarm */
         RmDiag  diagn;            /* Diagnostics (if any) */
      } usta;
   } t;
} RmMngmt;
```

hdr

Header structure. For a description, refer to Section 3.3.1, "Header."

cfm

The status field is not significant to this primitive.

alarm

Alarm. It contains the category, event, and cause of the alarm. These fields are described next.

```
typedef struct cmAlarm
{
   DateTime dt;      /* data and time */
   U16 category;     /* alarm category */
   U16 event;        /* alarm event */
   U16 cause;        /* alarm cause */
}CmAlarm;
```

dt

Date and time structure.

category

Tells to which category the error is related. Currently, only one category is supported.

Name	Description
LCM_CATEGORY_PROTOCOL	When an event occurred is protocol-related

event

Specifies the event that has occurred. The following categories are supported.

Name	Description
LRM_EVENT_TFCMTRC_CHANGED	The traffic metric parameter changed
LRT_EVENT_PAPAUD_SEQ	Out-of-sequence for PAP auditing
LRT_EVENT_OAPAUD_SEQ	Out-of-sequence for OAP auditing
LRT_EVENT_GAPAUD_SEQ	Out-of-sequence for GAP auditing
LRT_EVENT_PAPAUD_CFMTMR	Audit confirm timer expired for the PAP
LRT_EVENT_OAPAUD_CFMTMR	Audit confirm timer expired for the OAP
LRT_EVENT_GAPAUD_CFMTMR	Audit confirm timer expired for the GAP
LRT_EVENT_PAPAUD_PEORIDTMR	Period timer for auditing has expired
LRT_EVENT_PAPAUD_FINISHED	PAP auditing has finished
LRT_EVENT_OAPAUD_FINISHED	OAP auditing has finished
LRT_EVENT_GAPAUD_FINISHED	GAP auditing has finished

cause

Specifies the cause. The additional information in structure **RmDiag** depends on the cause.

Name	Description
LRM_CAUSE_AVCR_THRESHOLD_EXCEEDED	The threshold for the available cell rate. The traffic metric parameter exceeded.
LRT_CAUSE_AUD_CFM_OUTOFSEQENCE	Out-of-sequence for auditing
LRT_CAUSE_AUD_TMR_EXP	Auditing timer has expired.
LRT_CAUSE_AUD_FINISHED	Auditing has finished
LCM_REASON_MEM_NOAVAIL	Memory is not available in the system
LCM_REASON_INVALID_PAR_VAL	Invalid parameter type, such as the interface type or audit type

diagn

Diagnostics identifies the parameter that caused the request to fail. For more information, see **diagn** on page 203.

Description:

The RM uses this function to provide the layer manager with unsolicited status information (alarms). The unsolicited status can be enabled or disabled via the layer manager control request. The following alarms are generated by the RM.

Description	Category	Event	Cause
PNNI physical link traffic metric parameters are updated. The threshold for the available cell rate parameter has exceeded.	**LCM_CATEGORY_ PROTOCOL**	**LRM_EVENT_TFC MTRC_CHANGED**	**LRM_CAUSE_AVCR_ THRESHOLD_EXCEE DED**
PAP audit confirm timer has expired	**LCM_CATEGORY_ PROTOCOL**	**LLRM_EVENT_PA PAUD_CFMTMR**	**LRM_CAUSE_AUD_T MR_EXP**
PAP audit confirm is out-of-sequence	**LCM_CATEGORY_ PROTOCOL**	**LLRM_EVENT_PA PAUD_REQ**	**LRM_CAUSE_AUD_C FM_OUTOFSEQENCE**
PAP audit has finished	**LCM_CATEGORY_ PROTOCOL**	**LRM_EVENT_PAP AUD_FINISHED**	**LRM_CAUSE_AUD_F INISHED**
PAP period timer has expired	**LCM_CATEGORY_ PROTOCOL**	**LRM_EVENT_PAP AUD_PERIODTMR**	**LRM_CAUSE_AUD_T MR_EXP**
OAP audit confirm timer has expired	**LCM_CATEGORY_ PROTOCOL**	**LLRM_EVENT_OA PAUD_CFMTMR**	**LRM_CAUSE_AUD_T MR_EXP**

Description	Category	Event	Cause
OAP audit confirm is out-of-sequence	LCM_CATEGORY_PROTOCOL	LLRM_EVENT_OAPAUD_REQ	LRM_CAUSE_AUD_CFM_OUTOFSEQENCE
OAP audit has finished	LCM_CATEGORY_PROTOCOL	LRM_EVENT_OAPAUD_FINISHED	LRM_CAUSE_AUD_FINISHED
GAP audit confirm timer has expired	LCM_CATEGORY_PROTOCOL	LLRM_EVENT_GAPAUD_CFMTMR	LRM_CAUSE_AUD_TMR_EXP
GAP audit confirm is out-of-sequence	LCM_CATEGORY_PROTOCOL	LLRM_EVENT_GAPAUD_REQ	LRM_CAUSE_AUD_CFM_OUTOFSEQENCE
GAP audit has finished	LCM_CATEGORY_PROTOCOL	LRM_EVENT_GAPAUD_FINISHED	LRM_CAUSE_AUD_FINISHED
Out-of-memory in the system	LCM_CATEGORY_PROTOCOL	0	LCM_REASON_MEM_NOAVAIL
Invalid interface type	LCM_CATEGORY_PROTOCOL	interface type	LCM_REASON_INVALID_PAR_VAL

Returns:

00 ROK

01 RFAILED

3.7.2 Interface with the Upper Layers

The RM is the service provider for GCC and the PSIF.

The following primitives are provided at the interface between RM and GCC / PSIF, which is called the RMT interface.

Bind Establishment

Primitive Name	Description	Flow
`XxYyRmtBndReq`	Bind request	PSIF, GCC to RM
`XxYyRmtBndCfm`	Bind confirm	RM to PSIF, GCC

Resource Management Bind

Primitive Name	Description	Flow
`XxYyRmtAlocReq`	Resource allocation request	PSIF, GCC to RM
`XxYyRmtAlocCfm`	Resource allocation confirm	RM to GCC, PSIF
`XxYyRmtDalocReq`	Resource deallocation request	PSIF, GCC to RM
`XxYyRmtDalocCfm`	Resource deallocation confirm	RM to GCC
`XxYyRmtDalocInd`	Resource deallocation indication	RM to GCC
`XxYyRmtGrpAlocReq`	Resource group allocation request	PSIF to RM
`XxYyRmtGrpDalocReq`	Resource group deallocation request	PSIF to RM
`XxYyRmtAudReq`	Audit request	RM to PSIF, GCC
`XxYyRmtAudCfm`	Audit confirm	GCC, PSIF to RM
`XxYyRmtAudInd`	Audit indication	GCC to RM
`XxYyRmtAudRsp`	Audit response	RM to GCC

For a detailed description of the RMT interface, refer to the *Interworking Call Control Interface Service Definition*.

3.7.3 Interface with System Services

This section discusses RM's interface with system services.

3.7.3.1 General

This section describes the system services required by the RM.

Task Scheduling

The task scheduling management functions are called to register, activate, and terminate a task. Use the following functions for task scheduling management.

Name	Description
`SRegActvTsk`	Register activate task - Task
`rmActvTsk`	Activate task for the RM
`SPstTsk`	Post task
`SExitTsk`	Exit task

Initialization

OS calls the initialization management function to initialize a task. Use the following functions for initialization management.

Name	Description
`rmActvInit`	Activate task - Initialize the RM

Memory Management

The memory management functions allocate and deallocate variable-sized buffers utilizing static buffers. Use the following functions for memory management.

Name	Description
`SGetSBuf`	Get static buffer
`SGetSMem`	Get static memory

Message Management

Using dynamic buffers, the message management functions initialize, add data to and remove data from messages. Use the following functions for message management.

Name	Description
`SGetMsg`	Allocate a message (from a dynamic pool)
`SPutMsg`	Deallocate a message (into a dynamic pool)
`SFndLenMsg`	Find the length of a message
`SAddPreMsg`	Add an octet to the beginning of a message
`SAddPstMsg`	Add an octet to the end of a message
`SRemPreMsg`	Remove an octet from the beginning of a message
`SRemPstMsg`	Remove an octet from the end of a message
`SPkS8`	Add a signed 8-bit value to a message
`SPkU8`	Add an unsigned 8-bit value to a message
`SPkS16`	Add a signed 16-bit value to a message
`SPkU16`	Add an unsigned 16-bit value to a message
`SPkS32`	Add a signed 32-bit value to a message
`SPkU32`	Add an unsigned 32-bit value to a message
`SUnpkS8`	Remove a signed 8-bit value from a message
`SUnpkU8`	Remove an unsigned 8-bit value from a message
`SUnpkS16`	Remove a signed 16-bit value from a message
`SUnpkU16`	Remove an unsigned 16-bit value from a message
`SUnpkS32`	Remove a signed 32-bit value from a message
`SUnpkU32`	Remove an unsigned 32-bit value from a message

Timer Functions

The following timer functions are used.

Name	Description
`SRegTmr`	Register activation function - Timer
`SDeRegTmr`	Deregister activation function - Timer

Miscellaneous

Resource availability checking. The following miscellaneous functions are used:

Name	Description
`SFndProcId`	Find processor ID on which a task runs
`SGetDateTime`	Get real date and time
`SLogError`	Handle an error
`SPrint`	Print a preformatted string to the default display device

For a detailed description of the system services listed previously, refer to the *System Services Interface Service Definition*.

3.7.3.2 rmActvInit

Name:

Activate Task - Initialize the RM

Direction:

System services to the RM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 rmActvInit(ent, inst, region, reason)
Ent    ent;
Inst   inst;
Region region;
Reason reason;
```

Parameters:

`ent`

Entity ID.

`inst`

Instance ID for the entity.

`region`

Memory region ID that may be used by the layer to get static memory.

`reason`

Reason for initialization. Currently, this field is not used.

Description:

System services uses this function to initialize the RM.

Returns:

`00      ROK`

`01      RFAILED`

3.7.3.3 rmActvTsk

Name:

Activate Task

Direction:

System services to the RM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 rmActvTsk(pst, mBuf)
Pst    *pst;
Buffer *mBuf;
```

Parameters:

pst

For more description, refer to Section 3.3.3, "**Pst**."

mBuf

Message buffer.

Description:

System services uses this function, which injects an event and a primitive into the RM layer. The given message buffer is unpacked to find the corresponding primitive and associated parameters. Then, the appropriate primitive reception handler is scheduled.

Returns:

00 ROK

01 RFAILED

3.8 Switching Fabric Manager

This section discusses the Switching Fabric Manager (SFM) and details its interfaces and associated primitives.

3.8.1 Interface with the Layer Manager

This section discusses SFM’s interface with its layer manager (LSF).

3.8.1.1 Primitive Overview

Because the SFM is hardware-dependent, a dummy SFM is included in ICC. The code can be easily extended to support the underlying hardware. The SFM included in ICC consists of the basic TAPA framework for an entity that can have more than one upper SAP. The switching or deswitching request returns a confirmation without any check.

The primitives used between SFM and its layer manager are described in the following list.

Configuration

The following functions configure the protocol layer resources.

Name	Description
`SfMiLsfCfgReq`	Configuration request
`SfMiLsfCfgCfm`	Configuration confirm

Control

The following primitives can be used to control the SFM.

Name	Description
`SfMiLsfCntrlReq`	Control request
`SfMiLsfCntrlCfm`	Control confirm

Unsolicited Status

The SFM uses the following function to indicate status changes.

Name	Description
`SfMiLsfStaInd`	Status indication

3.8.1.2 Specific

This section details the primitives passed between SFM and its layer manager.

3.8.1.2.1 SfMiLsfCfgReq

Name:

Configuration Request

Direction:

Layer manager to the SFM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 SfMiLsfCfgReq(pst, cfg)
Pst     *pst;
SfMngmt *cfg;
```

Parameters:

pst

For more details, refer to Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. The configuration structure has the following format:

```
typedef struct sfMngmt
{
   Header    hdr;                 /* header */
   CmStatus  cfm;                 /* status in confirm */
   union
   {
/* configuration */

      struct
      {
         union
         {

            SfGenCfg  sfGen;      /* General Config */
            SfSapCfg  sfSap;      /* SAP Config */
          } s;
      } cfg;                      /* configuration */
   } t;
} SfMngmt;
```

hdr

For a description, refer to Section 3.3.1, "Header."

cfm

Status field. The confirmation primitives uses this field to report errors. It has no significance for the control request.

sfGen

General SFM configuration structure. The general configuration must be done first. The SFM uses much of the information carried by this table to reserve the necessary static memory.

```
typedef struct sfGenCfg          /* general configuration */
{
   U16       nmbSaps;            /* Number of Saps */
   SwtchIdx  maxSwtchIdx;        /* Maximum switch index to use */
   Pst       sm;                 /* Stack manager */
} SfGenCfg;
```

nmbSaps

Number of SAPs. This is the maximum number of SAPs toward GCC entities. The allowable values are: 1 to 32767.

maxSwtchIdx

Maximum number of the switch index. This is the number of switched resources that can exist at once. The allowable values are: 1 to 2^{32}-1.

sm

Post structure. It is used to communicate with the stack manager. The RM requires the post structure when sending unsolicited status. Unsolicited status is sent to the address in the **sm** field.

sfSapCfg

Upper SAP configuration structure. This SAP is used to communicate with GCC.

```
typedef struct sfSapCfg          /* SAP config */
{
   SpId      spId;               /* service provider id, SAP id */
   Priority  prior;              /* priority */
   Route     route;              /* route */
   Selector  selector;           /* selector */
   MemoryId  mem;                /* memory region & pool id */
} SfSapCfg;
```

spId

Service provider ID. The SFM uses this **spId** to identify the SAP on which it communicates with GCC.

prior

Priority used when the task buffers must be posted between the service provider and service user. It is used only in a loosely coupled system. The allowable value is:

PRIOR0 priority 0 - highest

route

The system uses this for internal routing requirements. TAPA does not define the contents or the use of this information. It is used only in a loosely coupled system. The allowable value is:

RTESPEC route to specific instance

selector

Defines whether the service provider and service user are loosely or tightly coupled. It is used to resolve a primitive call to the upper layer. The allowable values depend on the configuration. For more information, refer to the *Interworking Call Control Portation Guide*.

mem

For a description, see Section 3.3.5, "**Memory**."

Description:

The layer manager uses this function to configure the SFM. The general configuration must be done first.

Returns:

00 **ROK**

01 **RFAILED**

3.8.1.2.2 SfMiLsfCfgCfm

Name:

Configuration Confirm

Direction:

SFM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 SfMiLsfCfgCfm(pst, cfg)
Pst     *pst;
SfMngmt *cfg;
```

Parameters:

pst

For a description, refer to Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. With the exception of the following fields, the structure used for the configuration confirm is the same as that for the configuration request. See Section 3.8.1.2.1, "**SfMiLsfCfgReq**."

hdr

For a description, refer to Section 3.3.1, "Header."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
   U16  status;           /* Status of the operation  */
   U16  reason;           /* If failed, the reason    */
} CmStatus;
```

status

Indicates the status of the previous configuration request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Configuration request is successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible:

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_ELMNT	Invalid element specified in the configuration request
LCM_REASON_RECONFIG_FAIL	Failure in reconfiguration
LCM_REASON_MEM_NOAVAIL	Memory allocation failed
LCM_REASON_GENCFG_NOT_DONE	Configuration request received without previous general configuration
LCM_REASON_INVALID_SAP	Invalid SAP value passed (passed SAP does not exist in the system)
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header

Note: *The remaining fields are the same as those passed in the configuration request.*

Description:

The SFM uses this primitive to indicate to the layer manager the result of a configuration request.

Returns:

00 **ROK**

01 **RFAILED**

3.8.1.2.3 SfMiLsfCntrlReq

Name:

Control Request

Direction:

Layer manager to the SFM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 SfMiLsfCntrlReq(pst, cntrl)
Pst     *pst;
SfMngmt *cntrl;
```

Parameters:

pst

For a description, refer to Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. The control structure has the following format:

```
typedef struct sfMngmt
{
   Header    hdr;                  /* header */
   CmStatus  cfm;                  /* status in confirm */
   union
   {
/* Control */
      struct
      {
         DateTime dt;              /* date and time */
         U8 action;                /* action */
         U8 subAction;             /* sub action */
         union
         {
            U32   dbgMask;         /* debug mask */
         } c;
      } cntrl;                     /* control */
   }t;
} SfMngmt;
```

hdr

For a description, refer to Section 3.3.1, "Header."

cfm

This field is not significant to the control request.

dt

Date and time structure.

action

Specific action code. The allowable values are:

Name	Description
AENA	Enable
ADISIMM	Disable immediately

subAction

The allowable values are:

Name	Description
SAUSTA	Unsolicited status generation
SADBG	Debug option

dbgMask

Bit masks of different debug classes that can be enabled or disabled. This field specifies the classes of debug messages that must be controlled (enabled or disabled). The following debug classes are defined:

Name	Description
DBGMASK_UI	Upper interface debug information
DBGMASK_MI	Layer manager debug information

Description:

This function controls the SFM. The following table contains the possible operations with required parameters.

Description	subAction	action
Enable alarms	**SAUSTA**	**AENA**
Disable alarms		**ADISIMM**
Enable a debug class	**SADBG**	**AENA**
Disable a debug class		**ADISIMM**

Returns:

00 **ROK**

01 **RFAILED**

3.8.1.2.4 SfMiLsfCntrlCfm

Name:

Control Confirm

Direction:

SFM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 SfMiLsfCntrlCfm(pst, cntrl)
Pst     *pst;
SfMngmt *cntrl;
```

Parameters:

pst

For a description, refer to Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. With the exception of the fields described next, the structure used for the control confirm is the same as that for the control request. See Section 3.8.1.2.3, "**SfMiLsfCntrlReq**."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
    U16  status;              /* Status of the operation  */
    U16  reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure, this field contains the cause of it (**LCM_PRIM_NOK**). The following values are possible.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_INVALID_ACTION	Action passed in the control structure is not valid

Note: *The remaining fields are the same as those passed in the control request.*

Description:

The SFM uses this primitive to indicate to the layer manager the result of a control request.

Returns:

00 **ROK**

01 **RFAILED**

3.8.1.3 SfMiLsfStaInd

Name:

Status Indication

Direction:

SFM to the layer manager

Supplied:

In the layer manager

Synopsis:

```
PUBLIC S16 SfMiLsfStaInd(pst, sta)
Pst     *pst;
SfMngmt *sta;
```

Parameters:

pst

For a description, see Section 3.3.3, "**Pst**."

sta

Pointer to the status structure. The status structure has the following format:

```
typedef struct sfMngmt
{
   Header hdr;
   CmStatus  cfm;                  /* status in confirm */
   union
   {
/* unsolicited status */
      struct
      {
         DateTime dt;              /* date and time */
         CmAlarm  alarm;           /* alarm */
         union
         {
            SpId         spId;    /* service provider id */
         }t;
      } usta;
   } t;
} SfMngmt;
```

hdr

Header structure. For further information, see Section 3.3.1, "Header."

cfm

The status field is not significant to this primitive.

alarm

```
typedef struct cmAlarm
{
   DateTime dt;      /* data and time */
   U16 category;     /* alarm category*/
   U16 event;        /* alarm event */
   U16 cause;        /* alarm cause */
}CmAlarm;
```

dt

Date and time structure.

category

This field tells to which category the error is related. Currently, only one category is supported.

Name	Description
LCM_CATEGORY_INTERFACE	When an event is received on a SAP that is not configured or bound

event

This field specifies the event that has occurred. The supported categories include the following.

Name	Description
LCM_EVENT_INV_STATE	Invalid SAP state (SAP is not bound)
LCM_EVENT_UI_INV_EVT	Invalid event received from upper layer

cause

This field specifies the cause. Additional information in union **t** depends on the cause.

Name	Description
LCM_CAUSE_INV_SAP	Invalid SAP (the value that caused a problem is passed in the **spId** field)

spId

Service provider ID. Present only if the **cause** field identifies the SAP as invalid.

Description:

The SFM uses this function to provide the layer manager with unsolicited status information (alarms). The unsolicited status can be enabled or disabled via the layer manager control request. The SFM generates the following alarms.

Description	Category	Event	Cause
Invalid SAP ID received in the SFM primitives	**LCM_CATEGORY_INTERFACE**	**LCM_EVENT_UI_INV_EVT**	**LCM_CAUSE_INV_SAP**
SAP state associated with the SFM primitive is not bound	**LCM_CATEGORY_INTERFACE**	**LCM_EVENT_INV_STATE**	**LCM_CAUSE_INV_SAP**

Returns:

00 ROK

01 RFAILED

3.8.2 Interface with the Upper Layers

The SFM is the service provider for GCC.

The following primitives are provided at the interface between the SFM and GCC, which is called the SFT interface.

Bind Establishment

Primitive Name	Description	Flow
`XxYySftBndReq`	Bind request	GCC to SFM
`XxYySftBndCfm`	Bind confirm	SFM to GCC

Switching Establishment and Disestablishment

Primitive Name	Description	Flow
`XxYySftConReq`	Switching connect request	GCC to SFM
`XxYySftConCfm`	Switching connect confirm	SFM to GCC
`XxYySftRelReq`	Switching release request	GCC to SFM
`XxYySftRelCfm`	Switching release confirm	SFM to GCC
`XxYySftRelInd`	Switching release indication	SFM to GCC

For a detailed description of the SFT interface, refer to the *Interworking Call Control Interface Service Definition*.

3.8.3 Interface with System Services

This section discusses SFM’s interface with system services.

3.8.3.1 General

This section describes the system services required by the SFM.

Task Scheduling

The task scheduling management functions are called to register, activate, and terminate a task. Use the following functions for task scheduling management.

Name	Description
`SRegActvTsk`	Register activate task - Task
`sfActvTsk`	Activate task for the SFM
`SPstTsk`	Post task
`SExitTsk`	Exit task

Initialization

OS calls the initialization management function to initialize a task. Use the following function for initialization management.

Name	Description
`sfActvInit`	Activate task - Initialize the SFM

Memory Management

Using static buffers, the memory management functions allocate and deallocate variable-sized buffers. Use the following functions for memory management.

Name	Description
`SGetSBuf`	Get static buffer
`SGetSMem`	Get static memory

Message Management

The message management functions initialize, add data to, and remove data from messages utilizing dynamic buffers. Use the following functions for message management.

Name	Description
SGetMsg	Allocate a message (from a dynamic pool)
SPutMsg	Deallocate a message (into a dynamic pool)
SFndLenMsg	Find the length of a message
SAddPreMsg	Add an octet to the beginning of a message
SAddPstMsg	Add an octet to the end of a message
SRemPreMsg	Remove an octet from the beginning of a message
SRemPstMsg	Remove an octet from the end of a message
SPkS8	Add a signed 8-bit value to a message
SPkU8	Add an unsigned 8-bit value to a message
SPkS16	Add a signed 16-bit value to a message
SPkU16	Add an unsigned 16-bit value to a message
SPkS32	Add a signed 32-bit value to a message
SPkU32	Add an unsigned 32-bit value to a message
SUnpkS8	Remove a signed 8-bit value from a message
SUnpkU8	Remove an unsigned 8-bit value from a message
SUnpkS16	Remove a signed 16-bit value from a message
SUnpkU16	Remove an unsigned 16-bit value from a message
SUnpkS32	Remove a signed 32-bit value from a message
SUnpkU32	Remove an unsigned 32-bit value from a message

Miscellaneous

Resource availability checking. The following miscellaneous functions are used.

Name	Description
`SFndProcId`	Find a processor ID on which a task runs
`SGetDateTime`	Get real date and time
`SLogError`	Handle an error
`SPrint`	Print a preformatted string to the default display device

For a detailed description of the system services listed above, refer to the *System Services Interface Service Definition*.

3.8.3.2 sfActvInit

Name:

Activate Task - Initialize the SFM

Direction:

System services to the SFM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 sfActvInit(ent, inst, region, reason)
Ent    ent;
Inst   inst;
Region region;
Reason reason;
```

Parameters:

ent

Entity ID.

inst

Instance ID for the entity.

region

Memory region ID that may be used by the layer to get static memory.

reason

Reason for initialization. Currently, this field is not used.

Description:

System services uses this function to initialize the SFM.

Returns:

00 ROK

01 RFAILED

3.8.3.3 sfActvTsk

Name:

Activate Task

Direction:

System services to the SFM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 sfActvTsk(pst, mBuf)
Pst    *pst;
Buffer *mBuf;
```

Parameters:

`pst`

Destination post structure. For a description, see Section 3.3.3, "**`Pst`**."

`mBuf`

Message buffer.

Description:

System services uses this function, which injects an event and a primitive into the SFM layer. The given message buffer is unpacked to find the corresponding primitive and associated parameters. Then, the appropriate primitive reception handler is scheduled.

Returns:

`00      ROK`

`01      RFAILED`

3.9 Connection Manager

This section describes the Connection Manager (XM), discussing in detail its interfaces and associated primitives.

3.9.1 Interface with the Layer Manager

This section discusses the XM's interface with its layer manager (LXM).

3.9.1.1 Primitive Overview

The following list of primitives is used between the XM and its layer manager.

Configuration

This procedure configures the XM resources using the following functions.

Name	Description
`XmMiLxmCfgReq`	Configuration request
`XmMiLxmCfgCfm`	Configuration confirm

Control

This procedure activates and deactivates the XM resources using the following functions.

Name	Description
`XmMiLxmCntrlReq`	Control request
`XmMiLxmCntrlCfm`	Control confirm

Statistics

This retrieves the XM statistics information using the following functions.

Name	Description
`XmMiLxmStsReq`	Statistics request
`XmMiLxmStsCfm`	Statistics confirm

Solicited Status

This retrieves the status of XM using the following functions.

Name	Description
XmMiLxmStaReq	Status request
XmMiLxmStaCfm	Status confirm

Unsolicited Status

This indicates a change in the status of the XM using the following function.

Name	Description
XmMiLxmStaInd	Status indication

3.9.1.2 Specific

This section details the primitives used between the XM and its layer manager.

3.9.1.2.1 XmMiLxmCfgReq

Name:

Configuration Request

Direction:

Layer manager to the XM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 XmMiLxmCfgReq(pst, cfg)
Pst     *pst;
CcMngmt *cfg;
```

Parameters:

pst

For a description, refer to Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. The configuration structure has the following format:

```
typedef struct xmMngmt
{
   Header hdr;
   CmStatus cfm;
   /* Configuration */
      struct
      {
         union
         {
            XmGenCfg       xmGen;          /* XM Gen config */
            XmCcUpSapCfg   ccUpSapCfg;     /* Cc Upper Sap config */
            XmRmUpSapCfg   rmUpSapCfg;     /* Rm Upper Sap config */
            XmTkIwfCfg     tkIwfCfg;       /* Trunking IWF config */
            XmPh2TkIwfCfg  ph2TkIwfCfg;    /* Ph 2 trunking IWF cfg */
            XmSigVcciCfg   sigVcciCfg;     /* SIGVCCI Config */
            XmVcciCfg      vcciCfg;        /* VCCI config */
            XmCicCfg       cicCfg;         /* Ph 2 CIC Range cfg */
            XmPh2AtmRscCfg ph2AtmRscCfg;   /* ph2 ATM Resource cfg */
            XmAtmProfCfg   atmProfCfg;     /* ATM profile config */
            XmVtoaProfCfg  vtoaProfCfg;    /* VTOA profile config */
         }s;
      } cfg;
}XmMngmt;
```

hdr

Header structure. For further details, see Section 3.3.1, "Header."

cfm

Status field. For more information, refer to Section 3.3.2, "Status."

xmGen

General configuration structure. The general configuration must be done first. The XM uses much of the information carried by this structure to reserve the proper amount of static memory.

```
typedef struct xmGenCfg   /* General Configuration */
{
   U8  maxCcUpSap;        /* Max Numb of Conn. Mgmt upper SAPs  */
   U8  maxRmUpSap;        /* Max Numb of Res. Mgmt upper SAPs  */
   U8  maxCcInst;         /* Maximum Number of GCC instances */
   U32 maxTkCon;          /* Max Numb of ph 1 connections plus */
                          /* feature transparency connections */
   U32 maxPh2TkCon;       /* Max Numb of ph 2 connections */
   U16 maxTkIwf;          /* Max Numb of ph 1 trunking IWFs */
                          /* plus feature transparency IWFs */
                          /* configured  */
   U16 maxPh2TkIwf;       /* Max Numb of ph 2 IWFs configured */
   U16 maxPh2AtmRscCb;    /* Max Numb of ph 2 rsc control blk */
   U32 maxVCCI;           /* Max Number of AAL1/AAL2 VCCIs */
                          /* plus feature transparency VCCIs */
                          /* in the System */
   U32 maxCID;            /* Maximum Number of AAL1.AAL2 CIDs */
                          /* plus feature transparency CIDs in */
                          /* the System */
   U32 maxCic;            /* Maximum Number of CICs (for ph 2 */
                          /* trunking) in the System */
   U16 intfcIwfTblHlSz;   /* Interface IWF Table Hash List Size */
   U16 sctTblHlSz;        /* SCT Table Hash List Size */
   U16 spConnTblHlSz;     /* Maximum size of SpConnId hash list */
   U16 vcciTblHlSz;       /* VCCI Table Hash List Size */
   U16 atmRscTblHlSz;     /* ATM resource Table Hash List Size */
   U8  nmbAtmProfs;       /* number of ATM profiles */
   U8  nmbVtoaProfs;      /* number of VTOA profiles */
   U16 flowThreshUp;      /* maximum number of primitives that */
                          /* can be simultaneously sent to the */
                          /* upper layer */
   S16 timeRes;           /* time resolution */
   Pst sm;                /* Post Structure to Stack Manager */
} XmGenCfg;
```

maxCcUpSAP

Maximum number of connection management upper SAPs configured in the XM.

maxRmUpSAP

Maximum number of resource management upper SAPs configured in the XM.

maxCcInst

The highest call control ID (CID) among the CIDs of the GCC entities currently communicating with the XM, via the **CcUpSAP**s.

maxTkCon

The XM requires this information to reserve the static memory required for phase 1 trunking connection control blocks or feature transparency connection control blocks.

maxPh2TkCon

The XM requires this information to reserve the static memory required for phase 2 trunking connection control blocks.

maxTkIwf

The XM requires this information to reserve the static memory required for phase 1 trunking IWF control blocks or feature transparency IWF control blocks.

maxPh2TkIwf

The XM requires this information to reserve the static memory required for phase 2 trunking IWF control blocks.

maxPh2AtmRscCb

Maximum number of phase 2 AAL1/AAL2 ATM resource control blocks in the system.

maxVCCI

Maximum number of AAL1/AAL2/feature transparency VCCIs in the system.

maxCID

Maximum number of AAL1/AAL2/feature transparency CIDs in the system.

maxCic

Maximum number of phase 2 trunking CICs in the system.

intfcIwfTblHlSz

Size of the trunking IWF control block hash list. The ideal value is equal to the number of parallel IWF control blocks that exist in the system. In this case, each hash list bin has a maximum of one entry and the search time is minimal. By reducing the size of the hash list, the search time increases but less memory is required. There is always a trade-off between time and memory. A good value is about one fourth of the number of connections, so that a hash list bin has a maximum of four entries.

sctTblHlSz

Size of the SCT connection hash list. Refer to the description of **intfcIwfTblHlSz** on how to choose the value.

spConTblHlSz

Size of the **spConnId** connection hash list. Refer to the description of **intfcIwfTblHlSz** on how to choose the value.

vcciTblHlSz

Size of the VCCI control block hash list. Refer to the description of **intfcIwfTblHlSz** on how to choose the value.

atmRscTblHlSz

Size of the Phase 2 ATM resource control block hash list. Refer to the description of **intfcIwfTblHlSz** on how to choose the value.

nmbATMProfs

Number of ATM profiles in the system. ATM profiles contain the ATM connection parameters required to create the SETUP message for SVC establishment, for a signalling or a bearer VCCI. These parameters include the ATM traffic descriptor, AAL parameter, bearer capability, and QoS parameters. The **atmProfile** is required only in the case of AAL1/AAL2 ATM trunking.

nmbVtoaProf

Number of VTOA profiles in the system. The VTOA profile contains the AAL2 connection profile ID associated with an ATM bearer VCC. All CIDs on the bearer VCC use the same VTOA profile ID. The XM passes this profile ID, which is used during narrowband call switching, to GCC. The VTOA profile is required only in the case of AAL2 ATM trunking.

flowThresUp

This indicates the number of primitives XM sends to GCC simultaneously for a bulk operation on a VCC connection. For example, Aloc Confirm for all pending CIDs after the connection for the VCCI (to which the CIDs belong) is established; and the **DealocInd** for all active CIDs if the VCCI connection is released. In such situations, the XM sends **flowThreshUp** messages to GCC and starts the appropriate flow control timer—**SIGVCCFLC** for the signalling VCC or **BEARERVCCFLC** for bearer VCCs.

Upon expiration of this timer, the XM can re-send **flowThreshUp** messages (if applicable) and restart the timer. This process continues until all the pending messages for that VCC connection have been sent.

timeRes

Timer resolution, that is, the period during which the common timer function is called for this module. The module uses this period internally to maintain different timers for different connections.

sm

Post structure. It is used for communicating with the stack manager. The XM requires the post structure when sending unsolicited status. Unsolicited status is sent to the address in the **sm** field.

ccUpSapCfg

It is used to configure a connection management upper SAP in the system. This SAP is required for connection management communication between the XM and GCC.

```
typedef struct xmCcUpSapCfg     /* Conn. Mgmt Upper Configuration
structure */
{
   SpId spId;                   /* service provider id */
   Priority prior;              /* priority */
   Route route;                 /* route */
   Selector selector;           /* selector */
   MemoryId mem;                /* memory region & pool id */
} XmCcUpSapCfg;
```

spId

Service provider ID. The XM uses this **spId** to identify the SAP on which it communicates with GCC.

prior

Priority. It is used when the task buffers must be posted between the service provider and service user. It is used only in a loosely coupled system. The allowable value is:

PRIOR0 priority 0 - highest.

selector

Defines whether the service provider and service user are loosely or tightly coupled. It is used to resolve a primitive call to the upper layer. The allowable values depend on the configuration.

route

The system uses this for internal routing requirements. TAPA does not define the contents or use of this information. It is used only in a loosely coupled system. The allowable value is:

RTESPEC route to specific instance.

mem

For a description, see Section 3.3.5, "**Memory**."

rmUpSapCfg

It is used to configure a resource management upper SAP in the system. This SAP is required for resource management communication between the XM and GCC.

```
typedef struct xmRmUpSapCfg     /* Res. Mgmt Upper Configuration
structure */
{
   SpId spId;                   /* service provider id */
   Priority prior;              /* priority */
   Route route;                 /* route */
   Selector selector;           /* selector */
   MemoryId mem;                /* memory region & pool id */
} XmRmUpSapCfg;
```

spId

Service provider ID. The XM uses this **spId** to identify the SAP on which it communicates with GCC.

prior

Priority. It is used when the task buffers must be posted between the service provider and service user. It is used only in a loosely coupled system. The allowable value is:

PRIOR0 priority 0 - highest.

selector

Defines whether the service provider and service user are loosely or tightly coupled. It is used to resolve a primitive call to the upper layer. The allowable values depend on the configuration.

route

The system uses this for internal routing requirements. TAPA does not define the contents or use of this information. It is used only in a loosely coupled system. The allowable value is:

RTESPEC route to specific instance.

mem

For more description, refer to Section 3.3.5, "**Memory**."

tkIwfCfg

The configuration for a phase 1 AA1/AAL2 trunking or for a feature transparency trunking IWF control block.

```
typedef struct xmTkIwfCfg     /* Trunking IWF CB Config */
{
   RmInterface intfc;         /* trunking interface */
   U8          iwfType;       /* type of IWF */
   U8          iwfPrtclType;  /* IWF signalling protocol */
   union
   {
#if (XM_PH1_TK || XM_PH2_TK)
     AmCdPtyNmb atmAddr;       /* Address of the terminating IWF */
#endif /* (XM_PH1_TK || XM_PH2_TK) */
#ifdef XM_FEATTRP_SI
     SiCdPtyNum isupAddr;      /* Address of the terminating IWF */
#endif /* XM_FEATTRP_SI */
#ifdef XM_FEATTRP_IN
     CdPtyNmb   dss1Addr;      /* Address of the terminating IWF */
#endif /* XM_FEATTRP_IN */
     U8         pad;
   }termIwfAddr;
   union
   {
#if (XM_PH1_TK || XM_PH2_TK)
     AmCgPtyNmb atmAddr;       /* Address of the originating IWF */
#endif /* (XM_PH1_TK || XM_PH2_TK) */
#ifdef XM_FEATTRP_SI
     SiCgPtyNum isupAddr;      /* Address of the originating IWF */
#endif /* XM_FEATTRP_SI */
#ifdef XM_FEATTRP_IN
     CgPtyNmb   dss1Addr;      /* Address of the originating IWF */
#endif /* XM_FEATTRP_IN */
     U8         pad;
   }origIwfAddr;
    Vcci        minCntrlgVcci;  /* Lowest valid controlg VCCI */
   Vcci        maxCntrlgVcci;   /* Highest valid controlg VCCI */
   Vcci        minCntrldVcci;   /* Lowest valid controld VCCI */
   Vcci        maxCntrldVcci;   /* Highest valid controld VCCI */
   ProfId      defAtmProfile;   /* Deafault ATM profile ID */
   ProfId      defVTOAProfile;  /* Deafault VTOA profile ID */
   U8          defVcciType;     /* the type of SVC based VCCIs
                                /* to be allocated */
   U8          defMaxCID;       /* the number of CID per SVC
                                /* based VCCI */
   XmTmrCfg    tmr;             /* Connection Manager timers */
} XmTkIwfCfg;
```

intfc

The trunking interface to which the given IWF control block belongs. For more details, see Section 3.3.6, "Interface."

iwfType

Type of trunking IWF control block. The following values are allowed:

Value	Description
LXM_IWFTYPE_PH1_AAL1	AAL1 phase 1 trunking IWF
LXM_IWFTYPE_PH1_AAL2	AAL2 phase 1 trunking IWF
LXM_IWFTYPE_FEATTRP_SI	Feature transparency IWF using ISUP
LXM_IWFTYPE_FEATTRP_IN	Feature transparency IWF using DSS1
LXM_IWFTYPE_PH2_AAL1	AAL1 phase 2 trunking IWF
LXM_IWFTYPE_PH2_AAL2	AAL2 phase 2 trunking IWF

`iwfPrtclType`

The protocol type used to establish an SVC connection for the VCCIs of the trunking IWF control block. The **`iwfPrtclType`** can take the following values:

For the feature transparency IWFs:

Value	Description
`CC_SIITU92`	ISUP ITU 92 protocol variant
`CC_SIANS92`	ISUP ANSI 92 protocol variant
`CC_INITU`	ISDN ITU
`CC_INETSI`	ISDN ETSI

For the AAL1 / AAL2 trunking IWFs:

Value	Description
`CC_AM_SIG_PNNI`	Q.93B PNNI protocol variant
`CC_AM_Q2931`	Q.93B Q.2931 protocol variant
`CC_AM_UNI40`	Q.93B UNI40 protocol variant
`CC_AM_UNI31`	Q.93B UNI3.1 protocol variant

`termIwfAddr`

This is the address of the remote IWF associated with this IWF control block. The address can be an ATM, ISUP, or ISDN (DSS1) address.

`origIwfAddr`

This is the address of the local IWF associated with this IWF control block. The address can be an ATM, ISUP, or ISDN (DSS1) address.

`minCntrlgVcci`

This is the minimum valid controlling VCCI configured for this IWF control block.

`maxCntrlgVcci`

This is the maximum valid controlling VCCI configured for this IWF control block.

`minCntrldVcci`

This is the minimum valid controlled VCCI configured for this IWF control block.

`maxCntrldVcci`

This is the maximum valid controlled VCCI configured for this IWF control block.

defAtmProfile

This is the ID of the default ATM profile used to obtain the ATM connection parameters required for SVC connection establishment of all the bearer VCCs, which belong to this IWF. Some ATM profiles can be configured in the system, and the **defAtmProfile** is an index in the list of ATM profiles.

defVtoaProfile

This is the ID of the default VTOA profile used for the AAL2 connections, which belong to this IWF control block. Some VTOA profiles can be configured in the system, and the **defVtoaProfile** is an index in the list of VTOA profiles.

defVcciType

The default VCCI type used when an SVC connection is established for a new VCCI, on that particular IWF. The allowable values are:

Value	Description
LXM_BEARERVCCI_AAL2	AAL2 bearer VCCI
LXM_BEARERVCCI_AAL1_SINGLE	One-to-one AAL1 bearer
LXM_BEARERVCCI_AAL1_MUX	Many-to-one AAL1 bearer
LXM_VCCI_FEATTRP	Feature transparency VCCI

defMaxCID

The default number of CIDs defined when an SVC connection is set up for a new VCCI on the given trunking IWF control block. The allowable values: 0 to 255.

tmr

For information on the timer configuration, see Section 3.3.4, "Timer Configuration."

```
typedef struct xmTmrCfg
{
   TmrCfg tSIGVCCRLS;     tSIGVCCRLS timer - signalling
                          VCC connection release timer
   TmrCfg tBEARERVCCRLS;  tBEARERVCCRLS timer - bearer
                          VCC connection release timer
   TmrCfg tVCCFLC;        tVCCFLC timer - for release of CIDs
                          for a signalling/bearer VCC con
} XmTmrCfg;
```

tSIGVCCRLS

This timer can be configured in the XM to release a signalling VCC connection after a wait period.

tBEARERVCCRLS

This timer can be configured in the XM to release a bearer VCC connection after a wait period.

tVCCFLC

Flow control timer for a VCC connection. This timer is used when the XM sends primitives to GCC simultaneously for a bulk operation on a VCC connection. For example, Aloc Confirm for all pending CIDs after the connection for the VCCI (to which the CIDs belong) is established; and **DealocInd** for all active CIDs if the VCCI connection is released.

In such situations, the XM sends **flowThreshUp** (refer to the XM general configuration) messages to GCC and starts the flow control timer. At expiration of this timer, the XM can re-send **flowThreshUp** messages (if applicable) and restart the timer. This process continues until all pending messages for the VCC connection are sent.

ph2TkIwfCfg

It is used to configure a phase 2 AAL1/AAL2 trunking IWF block in the system. An IWF control block represents an interface between a pair of originating and terminating IWFs.

```
typedef struct xmPh2TkIwfCfg   /* AAL1/AAL2 phase2 trunking IWF CB
Config */
{
   RmInterface intfc;         /* trunking interface */
   U8          iwfType;       /* IWF type (AAL1/AAL2 */
   U8          alocMeth;      /* CIC Allocation Method to be used */
   U16         maxCic;        /* Highest CIC configured on this DPC */
} XmPh2TkIwfCfg;
```

intfc

The ISUP trunking interface to which the given IWF block belongs. For further details, see Section 3.3.6, "Interface."

alocMeth

The method by which the phase 2 trunking CICs (circuits) should be allocated. The following methods are available:

- Highest available: The highest available CIC is selected.
- Lowest available: The lowest available CIC is selected.
- ITU-T method 2 (Q.764): As specified in Q.764, each node of a bothway circuit group has priority access to the group of circuits that it controls. Each node controls one half of the circuits in a bothway circuit group. The node with the higher signalling point code controls all even-numbered circuits (CIC) and the other node controls the odd-numbered circuits.

Within each group, the circuit that has been released the longest is selected (first-in, first-out). Each node of a bothway circuit group has non-priority access to the group of circuits that it does not control. Of this group, the latest released circuit is selected (last-in, first-out) if all the circuits in the group are busy.

The allowable values are:

Value	Description
LXM_AM_LOWEST_AVAIL	Lowest CIC available
LXM_AM_HIGHEST_AVAIL	Highest CIC available
LXM_AM_ITU_MTHD2	ITU method 2 allocation

iwfType

Type of trunking IWF control block. The following values are allowed:

Value	Description
LXM_IWFTYPE_PH1_AAL1	AAL1 phase 1 trunking IWF
LXM_IWFTYPE_PH1_AAL2	AAL2 phase 1 trunking IWF
LXM_IWFTYPE_FEATTRP_SI	Feature transparency IWF using ISUP
LXM_IWFTYPE_FEATTRP_IN	Feature transparency IWF using DSS1
LXM_IWFTYPE_PH2_AAL1	AAL1 phase 2 trunking IWF
LXM_IWFTYPE_PH2_AAL2	AAL2 phase 2 trunking IWF

maxCic

The highest available CIC configured at this interface.

sigVcciCfg

It is used to configure an AAL1/AAL2 signalling VCCI for an AAL1/AAL2 trunking IWF block.

```
typedef struct xmSigVcciCfg /* AAL2 SigVCCI Config */
{
   RmInterface intfc;         /* trunking intfc */
   Vcci        vcci;          /* vcci */
   U8          vcciType;      /* type of VCCI */
   Bool        isItCntrlg;    /* VCCI controlling/controlled? */
   Bool        isPVC;         /* Is VCCI PVC based? */
   ProfId      atmProfile;    /* the ATM profile associated */
   U8          state;         /* initial State of VCCI */
   U8          appId;         /* application identifier */
} XmSigVcciCfg;
```

intfc

The trunking interface to which the given VCCI belongs. For more details, see Section 3.3.6, "Interface."

vcci

The AAL1 / AAL2 signalling VCCI value.

vcciType

The type of VCCI. This can take one of the following values.

Value	Description
LXM_SIGVCCI_OVERAAL5	Signalling VCCI over AAL5
LXM_SIGVCCI_OVERAAL2	Signalling VCCI over AAL2

isItCntrlg

This is the signalling VCCI, a controlled VCCI, or a controlling VCCI.

isPVC

This is the flag if the signalling VCC connection is PVC-based or SVC-based.

atmProfile

This is the ID of the default ATM profile used to obtain the ATM connection parameters required for connecting SVCs for this VCCI.

state

State of the VCCI. The initial state of a VCCI can be available (**TRUE**) or not available (**FALSE**) for allocation.

appId

The type of narrowband protocol trunked on this signalling VCC.

vcciCfg

This is used to configure a PVC-based AAL1 / AAL2 VCCI for an AAL1 / AAL2 trunking IWF block.

```
typedef struct xmVcciCfg      /* AAL2/Many-to-one AAL1 VCCI Config */
{
   RmInterface  intfc;         /* trunking intfc */
   Vcci         vcci;          /* vcci */
   U8           vcciType;      /* type of VCCI */
   CID          maxCID;        /* Maximum Valid CID Value */
   Bool         isItCntrlg;    /* VCCI controlling/controlled? */
   ProfId       atmProfile;    /* the ATM profile associated */
   U8           state;         /* state of VCCI */
   RmRsc        trnkRsc;       /* associated trunking resource */
} XmVcciCfg;
```

intfc

The trunking interface to which the given VCCI belongs. For more details, see Section 3.3.6, "Interface."

vcci

The VCCI value.

vcciType

The type of VCCI. It can take one of the following values.

Value	Description
LXM_BEARERVCCI_AAL2	AAL2 bearer VCCI
LXM_BEARERVCCI_AAL1_MUX	Many-to-one AAL1 bearer VCCI

minCID

This is the minimum valid CID configured for this VCCI control block.

maxCID

This is the maximum valid CID configured for this VCCI control block.

atmProfile

This is the ID of the default ATM profile used to obtain the ATM connection parameters, which is required for establishing the SVC connection of this VCCI. The **atmProfile** is valid only if the **vcciType** indicates that it is a signalling VCCI. For bearer VCCIs, the default ATM profile associated with the IWF control block to which the given VCCI belongs is used.

isItCntrlg

This is the VCCI, a controlled VCCI, or a controlling VCCI. Depending on the value of this flag, the VCCI control block is inserted in the controlling VCCI-free queue or the controlled VCCI-free queue.

state

State of the VCCI. The state of a VCCI can be available (**TRUE**) or not available (**FALSE**) for allocation.

trnkRsc

Associated trunking resource. It contains the VPI/VCI of the PVC used as a VCCI.

cicCfg

It is used to configure a set of CICs for a phase 2 trunking IWF block.

```
typedef struct xmCicCfg /* Narrowband CIC Range Configuration */
{
   RmInterface  intfc; /* trunking intfc to which this CIC belongs */
   U16  strtCic;       /* Starting CIC */
   U16  numCic;        /* number of CIC Configured */
   U8   cicType;       /* type of CIC */
   U8   cntld;         /* Which CIC's are Controlled by other node */
   U8   state;         /* Initial state of the CIC */
} XmCicCfg;
```

intfc

The trunking interface to which the given CIC belongs. For a description, see Section 3.3.6, "Interface."

strtCic

Starting CIC configured at this interface.

numCic

Number of CICs configured, starting from the CIC specified in the **strtCic**.

cicType

Identifies whether the CICs can be used for incoming, outgoing, or bothway calls. The allowable values are:

Value	Description
LXM_CIC_OUTGOING	Outgoing calls allowed
LXM_CIC_INCOMING	Incoming calls allowed
LXM_CIC_BOTHWAY	Bothway = INCOMING | OUTGOING

cntld

Specifies which circuits are controlled by the remote node. This field is used for ITU method 2 allocation. By default, all circuits are assumed to be in the controlling list of this node. This field contains a bit mask. The least significant bit (LSB, Bit 0) indicates that the odd circuits are controlled by the remote node. Bit 1 indicates that the even circuits are controlled by the remote node. If all the circuits are controlled by the remote node, both flags must be set.

Value	Description
LXM_CNTRLD_ODD	Odd circuits are controlled
LXM_CNTRLD_EVEN	Even circuits are controlled
LXM_CNTRLD_ALL	All circuits (ODD|EVEN)

state

State of the circuit. The state of a circuit can be available (**TRUE**) or not available (**FALSE**) for allocation.

ph2AtmRscCfg

Phase 2 ATM resource configuration. The phase 2 ATM resource control block contains information on AAL1/AAL2 resources, which are used to set up the bearer connection for phase 2 trunking calls toward a remote phase 2 IWF. These phase 2 ATM resource control blocks are maintained in a hash list indexed on the remote IWF address.

The structure of the phase 2 ATM resource is similar to the **XmPh1TkIwfCfg**. Refer to the description of the **ph1TkIwfCfg** configuration for a detailed description of the **xmPh2AtmRscCfg** fields.

atmProfCfg

The ATM profile ID containing the ATM connection parameters required to create the SVC connection SETUP message for the signalling or bearer VCCIs for AAL1/AAL2 trunking. The **atmProfile** should be configured before **sigVcci** configuration.

For a detailed description of the previous data structure, see the broadband profile configuration of GCC (**CcMiLccCfgReq**).

vtoaProfCfg

VTOA profile configuration. The AAL2 trunking specification (ref) defines some AAL2 VTOA profiles dynamically associated with the AAL2 voice/voiceband data connection. The VTOA profile structure contains the ID of these AAL2 profiles. The actual profiles are defined in the SFM, and the XM passes the VTOA profile ID associated with an AAL2 connection to the GCC when the resource is allocated.

Some VTOA profiles can be configured in the XM. The XM maintains the following information for each of the profiles.

```
typedef struct xmVtoaProfCfg /* ATM Profile Configuration
                              Structure */
{
   U8 profId;              /* profile identifier */
   U8 aal2profId;          /* AAL2 profile identifier */
   U8 TMR;                 /* transmission medium requirement */
} XmVtoaProfCfg;
```

profId

Profile ID of the VTOA profile in the XM. The allowable values are: 0 to 255.

aal2ProfId

The AAL2 profile ID associated with this VTOA profile. This is the profile ID as defined in the AAL2 trunking document. The allowable values are: 0 to 15.

TMR

The narrowband transmission medium requirement with which the given VTOA profile is associated.

Description:

The layer manager uses this function to configure the XM.

Returns:

00 ROK

01 RFAILED

3.9.1.2.2 XmMiLxmCfgCfm

Name:

Configuration Confirm

Direction:

XM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 XmMiLxmCfgCfm(pst, cfg)
Pst     *pst;
CcMngmt *cfg;
```

Parameters:

pst

For more description, see Section 3.3.3, "**Pst**."

cfg

Pointer to the configuration structure. Except for the following fields, the structure used for the configuration confirm is the same as that for the configuration request. For a description, see Section 3.5.1.2.1, "**CcMiLccCfgReq**."

cfm

The status field has the following format:

```
typedef struct cmStatus
{
    U16   status;              /* Status of the operation  */
    U16   reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous configuration request primitive. It contains one of the following values:

Name	Description
LCM_PRIM_OK	Configuration request successful
LCM_PRIM_NOK	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
LCM_REASON_MEM_NOAVAIL	Memory allocation failed
LCM_REASON_REGTMR_FAIL	**SRegTmr** returned failure
LCM_REASON_GENCFG_NOT_DONE	Configuration request received without previous general configuration
LCM_REASON_EXCEED_CONF_VAL	Maximum value as given in the general configuration is exceeded. For example, the layer manager tries to configure SAP 5 but the maximum number of SAPs passed in the general configuration is 4.
LCM_REASON_RECONFIG_FAIL	Failure in reconfiguration
LCM_REASON_INVALID_SAP	Invalid SAP value passed. The passed SAP does not exist in the system.
LCM_REASON_HASHING_FAILED	Hash list library returned failure
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header

Description:

This function returns the result of a configuration request.

Returns:

00 ROK

01 RFAILED

3.9.1.2.3 XmMiLxmCntrlReq

Name:

Control Request

Direction:

Layer manager to the XM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 XmMiLxmCntrlReq(pst, cntrl)
Pst     *pst;
CcMngmt *cntrl;
```

Parameters:

pst

For a description, refer to Section 3.3.3, "**Pst**."

cntrl

Pointer to the control structure. The control structure has the following format.

```
typedef struct xmMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
/* Control */
      struct
      {
         DateTime dt;              /* date and Time */
         U8 action;                /* Action */
         U8 subAction;             /* SubAction */
         union
         {
            XmCcUpSapCntrl   ccUpSapCntrl;   /* xm upper sap control */
            XmRmUpSapCntrl   rmUpSapCntrl;   /* rm upper sap control */
            XmTkIwfCntrl     tkIwfCntrl;     /* trunking IWF control */
            XmPh2TkIwfCntrl  ph2TkIwfCntrl;  /* ph 2 trunking IWF cntrl */
            XmVcciCntrl      vcciCntrl;      /* VCCI control */
            XmCIDCntrl       cidCntrl;       /* CID control */
            XmCicCntrl       cicCntrl;       /* Narrowband CIC control */
            XmAtmProfCntrl   atmProfCntrl;   /* ATM profile Control */
            XmVtoaProfCntrl  vtoaProfCntrl;  /* VTOA profile Control */
            XmSigVcciCntrl   sigVcciCntrl;   /* SIGVCCI control */
            XmPh2AtmRscCntrl ph2AtmRscCntrl; /* ph2 ATM Resource Control*/
            U32              dbgMask;        /* debug mask */
         }s;
      } cntrl;
   } t;
} XmMngmt;
```

hdr

For more description, see Section 3.3.1, "Header."

cfm

It is used only in confirm primitives.

dt

Date and time structure.

subAction

Type of control procedure requested. The allowable values are:

Name	Description
SAUSTA	Unsolicited status generation
SADBG	Trace generation
SAELMNT	Specific element

action

Specific action code. The allowable values are:

Name	Description
AENA	Enable
ADISIMM	Disable immediately
ADEL	Delete
ARST	Reset

ccUpSapCntrl

The control structure for a trunking interface.

```
typedef struct xmCcUpSapCntrl
{
   SpId spId;
} XmCcUpSapCntrl;
```

spId

Upper SAP ID.

rmUpSapCntrl

The control structure for a trunking interface.

```
typedef struct xmRmUpSapCntrl
{
   SpId spId;
} XmRmUpSapCntrl;
```

spId

Upper SAP ID.

tkIwfCntrl

The control structure for a trunking interface.

```
typedef struct xmTkIwfCntrl     /* trunking IWF CB Control Structure */
{
   RmInterface  intfc;/* trunking interface to which this IWF belongs */
} XmTkIwfCntrl;
```

intfc

Interface to which the trunking IWF control block belongs. For more details, see Section 3.3.6, "Interface."

ph2TkIwfCntrl

The control structure for a trunking interface is:

```
typedef struct xmPh2TkIwfCntrl   /* phase 2 trunking */
{
   RmInterface intfc;         /* trunking interface */
} XmPh2TkIwfCntrl;
```

intfc

Interface to which the trunking IWF control block belongs. For more details, see Section 3.3.6, "Interface."

vcciCntrl

The control structure for a VCCI on an AAL1/AAL2 trunking interface.

```
typedef struct xmVcciCntrl
{
   RmInterface  intfc;       /* trunking interface */
   Vcci   vcci;              /* VCCI */
} XmVcciCntrl;
```

intfc

Interface to which the VCCI control block belongs. For more details, see Section 3.3.6, "Interface."

vcci

The VCCI ID of the affected VCCI control block.

CIDCntrl

The control structure for a CID control block.

```
typedef struct XmCIDCntrl
{
   RmInterface  intfc;       /* trunking interface */
   Vcci   vcci;              /* VCCI */
   CID    cid;                /* CID */
} XmCIDCntrl;
```

intfc

Interface to which the CID control block belongs. For a description, refer to Section 3.3.6, "Interface."

vcci

The VCCI ID of the VCCI control block that contains the CID.

cid

CID of the affected CID control block.

cicCntrl

The control structure for a CIC control block.

```
typedef struct rmCicCntrl
{
   RmInterface  intfc;       /* trunking interface */
   U16    cic;               /* Starting CIC */
   U16    numCic;            /* Number of CIC */
} XmCicCntrl;
```

intfc

Interface to which the CIC control block belongs. For a description, see Section 3.3.6, "Interface."

cic

CIC value. The control request affects all the circuits beginning with this start CIC.

numCic

Number of circuits affected by the control request. The number of circuits includes the start circuit.

atmProfCntrl

The control structure for an ATM profile.

```
typedef struct xmAtmProfCntrl
{
   U8   profId;  /* Profile Identifier */
} XmAtmProfCntrl;
```

profId

ATM profile ID.

vtoaProfCntrl

The control structure for a VTOA profile.

```
typedef struct xmVtoaProfCntrl
{
   U8   profId;  /* Profile Identifier */
} XmVtoaProfCntrl;
```

profId

VTOA profile ID.

sigVcciCntrl

It is used for enabling, disabling, or deleting a PVC / SVC-based signalling connection.

```
typedef struct xmSigVcciCntrl
{
   RmInterface intfc;        /* trunking interface */
   Vcci        vcci;         /* VCCI */
} XmSigVcciCntrl;
```

intfc

Interface to which the VCCI belongs. For more details, refer to Section 3.3.6, "Interface."

vcci

VCCI ID.

ph2AtmRscCntrl

It is used for enabling, disabling, or deleting a phase 2 trunking ATM resource control block.

```
typedef struct xmPh2AtmRscCntrl  /* phase 2 ATM Resource CB */
{
   RmInterface intfc;         /* trunking interface */
} XmPh2AtmRscCntrl;
```

`intfc`

Interface of the phase 2 trunking resource. For more information, see Section 3.3.6, "Interface."

`dbgMask`

Bit mask of different debug classes that can be enabled or disabled. This specifies the classes of debug messages that must be controlled (enabled or disabled). The following debug class is defined.

Name	**Description**
`DBGMASK_XM`	Internal XM debug class

Description:

This function controls the XM. The possible operations with required parameters are listed in the following table.

Description	**`subAction`**	**`action`**	**`elmnt`**	**others**
Enable alarms	**`SAUSTA`**	**`AENA`**	N / A	N / A
Disable alarms		**`ADISIMM`**		
Enable a debug class	**`SADBG`**	**`AENA`**		**`dbgMask`**
Disable a debug class		**`ADISIMM`**		
Delete a CC SAP toward GCC	**`SAELMNT`**	**`ADEL`**	**`STXMCCUPSAP`**	**`ccUpSapCntrl`**
Delete an RM SAP toward GCC		**`ADEL`**	**`STXMRMUPSAP`**	**`rmUpSapCntrl`**
Delete a trunking IWF control block		**`ADEL`**	**`STXMTKIWF`**	**`tkIwfCntrl`**
Delete a phase 2 trunking IWF control block		**`ADEL`**	**`STXMPH2TKIWF`**	**`ph2TkIwfCntrl`**
Delete a phase 2 ATM resource control block		**`ADEL`**	**`STXMPH2ATMRSC`**	**`ph2AtmRscCntrl`**
Delete a VCCI		**`ADEL`**	**`STXMVCCI`**	**`vcciCntrl`**
Make a VCCI available for allocation		**`AENA`**		
Make a VCCI unavailable for allocation		**`ADISIMM`**		
Delete a signalling VCCI		**`ADEL`**	**`STXMSIGVCCI`**	**`sigVcciCntrl`**

Description	**subAction**	**action**	**elmnt**	**others**
Make a signalling VCCI available for allocation		**AENA**		
Make a signalling VCCI unavailable for allocation		**ADISIMM**		
Delete a CIC		**ADEL**	**STXMCIC**	**cicCntrl**
Make a CIC available for allocation		**AENA**		
Make a CIC unavailable for allocation		**ADISIMM**		
Free a CIC. Remove the allocated status.		**ARST**		
Free a CID. Clear the call on the CID		**ARST**	**STXMCID**	**cidCntrl**
Delete an ATM profile		**ADEL**	**STXMATMPPROF**	**atmProfCntrl**
Delete a VTOA profile		**ADEL**	**STXMVTOAPPROF**	**vtoaProfCntrl**

Returns:

00 **ROK**

01 **RFAILED**

3.9.1.2.4 XmMiLxmCntrlCfm

Name:

Control Confirm

Direction:

XM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 XmMiLxmCntrlCfm(pst, cntrl)
Pst     *pst;
CcMngmt *cntrl;
```

Parameters:

`pst`

For a description, refer to Section 3.3.3, "**`Pst`**."

`cntrl`

Pointer to the control structure. Except for the following fields, the structure used for the control confirm is the same as that for the control request. For more details, refer to Section 3.5.1.2.3, "**`CcMiLccCntrlReq`**."

`cfm`

The status field indicates the result of a request. The status field has the following format.

```
typedef struct cmStatus
{
    U16   status;              /* Status of the operation  */
    U16   reason;              /* If failed, the reason    */
} CmStatus;
```

`status`

This field indicates the status of the previous control request primitive. It contains one of the following values:

Name	Description
`LCM_PRIM_OK`	Configuration request successful
`LCM_PRIM_NOK`	Configuration request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The remaining fields are the same as those passed in the control request. For further information, see Section 3.5.1.2.3. The following values are possible:

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCM_REASON_INVALID_ACTION	Invalid action passed in the control structure
LCM_REASON_INVALID_SUBACTION	Invalid subaction passed in the control structure

3.9.1.2.5 XmMiLxmStsReq

Name:

Statistics Request

Direction:

Layer manager to the XM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 XmMiLxmStsReq(pst, action, sts)
Pst     *pst;
Action  action;
CcMngmt *sts;
```

Parameters:

`pst`

For a description, refer to Section 3.3.3, "**`Pst`**."

`action`

Action indicator. The allowable values are:

Value	Description
`0`	Zero statistics counters (**`ZEROSTS`**)
`1`	Do not set the statistics counters to zero (**`NOZEROSTS`**)

sts

Pointer to the statistics structure. It has the following format:

```
typedef struct xmMngmt
{
   Header hdr;
   union
   {
/* Statistics */
      struct
      {
         DateTime dt;            /* Date and Time */
         union
         {
            XmTkIwfSts    tkIwfSts;    /* trunking */
            XmPh2TkIwfSts ph2TkIwfSts; /* AAL1/AAL2 ph 2 */
         } s;
      }sts;

   } t;
} XmMngmt;
```

hdr

For details, see Section 3.3.1, "Header."

The type of statistics information desired can be selected by programming the header substructure as:

elmnt

Element. The allowable values are:

Name	Description
STXMFEATTRPIWF	Feature transparency IWF statistics
STXMPH1TKIWF	Phase 1 trunking IWF statistics
STXMPH2TKIWF	Phase 2 trunking IWF statistics

dt

Date and time structure.

dura

Duration structure.

tkIwfSts

IWF statistics. The statistics confirm primitive returns the values. The **intfc** field specifies the interface to which the phase 1 trunking / feature transparency IWF belongs (for which statistics are requested). The other fields are explained in the statistics confirm primitive. For more information, refer to Section 3.5.1.2.6, "**CcMiLccStsCfm**."

```
typedef struct xmTkIwfSts /* Feature Trans. IWF Statistics */
{
   RmInterface intfc; /* trunking interface to which this IWF belongs */
   U32 alocReq;
   U32 alocSucc;
} XmTkIwfSts;
```

ph2TkIwfSts

The phase 2 trunking IWF statistics. The statistics confirm primitive returns the values. The **intfc** field specifies the interface to which the phase 2 trunking IWF belongs (for which the statistics are requested). The other fields are explained in the statistics confirm primitive. For details, see Section 3.5.1.2.6, "**CcMiLccStsCfm**."

```
typedef struct xmPh2TkIwfSts /* AAL1/AAL2 ph 2 trunking IWF stats */
{
   RmInterface intfc; /* trunking interface to which this IWF belongs */
   U32 alocReq;
   U32 alocSucc;
} XmPh2TkIwfSts;
```

Description:

The layer manager uses this function to gather statistics information from the XM.

Returns:

00 **ROK**

01 **RFAILED**

3.9.1.2.6 XmMiLxmStsCfm

Name:

Statistics Confirm

Direction:

XM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 XmMiLxmStsCfm(pst, sts)
Pst     *pst;
CcMngmt *sts;
```

Parameters:

pst

For a description, refer to Section 3.3.3, "**Pst**."

sts

Pointer to the statistics structure. The statistics structure has the following format.

```
typedef struct ccMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
/* Statistics */
      struct
      {
         DateTime dt;            /* Date and Time */
         union
         {
            XmTkIwfSts    tkIwfSts;    /* trunking */
            XmPh2TkIwfSts ph2TkIwfSts; /* AAL1/AAL2 ph 2 */
         } s;
      }sts;
    } t;
} XmMngmt;
```

hdr

Header structure. For more information, refer to Section 3.3.1, "Header."

cfm

Status field. The status field indicates the result of a request. It has the following format.

```
typedef struct cmStatus
{
   U16  status;              /* Status of the operation  */
   U16  reason;              /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values.

Name	Description
LCM_PRIM_OK	Statistics request successful
LCM_PRIM_NOK	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
LCM_REASON_INVALID_ENTITY	Invalid entity passed in the header
LCM_REASON_INVALID_INSTANCE	Invalid instance passed in the header
LCM_REASON_INVALID_MSGTYPE	Invalid message type passed in the header
LCC_REASON_INVALID_SAP	Invalid SAP specified

dt

Date and time structure.

dura

Duration structure.

tkIwfSts

IWF statistics. This structure is defined as:

```
typedef struct xmTkIwfSts /* Feature Trans. IWF Statistics */
{
   RmInterface intfc; /* trunking interface to which this IWF belongs */
   U32 alocReq;
   U32 alocSucc;
} XmTkIwfSts;
```

intfc

The interface to which the phase 1 trunking/feature transparency IWF control block belongs.

alocReq

Number of connections requested on this IWF block.

alocSucc

Number of connections successfully completed on this IWF block.

ph2TkIwfSts

The phase 2 trunking IWF statistics. This structure is defined as:

```
typedef struct xmPh2TkIwfSts /* AAL1/AAL2 ph 2 trunking IWF stats */
{
   RmInterface intfc; /* trunking interface to which this IWF belongs */
   U32 alocReq;
   U32 alocSucc;
} XmPh2TkIwfSts;
```

intfc

The interface to which the phase 2 trunking IWF control block belongs.

alocReq

Number of connections requested on this IWF block.

alocSucc

Number of connections successfully completed on this IWF block.

Description:

The XM uses this function to provide the layer manager with statistics information.

Returns:

00 ROK

01 RFAILED

3.9.1.2.7 XmMiLxmStaReq

Name:

Status Request

Direction:

Layer manager to the XM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 XmMiLxmStaReq(pst, sta)
Pst      *pst;
XmMngmt  *sta;
```

Parameters:

`pst`

For a description, refer to Section 3.3.3, "**`Pst`**."

sta

Pointer to the solicited status structure. The solicited status structure has the following format.

```
typedef struct xmMngmt
{
   Header hdr;
   CmStatus  cfm;                  /* status in confirm */
   union
   {
/* Solicited Status */
     struct
      {
         DateTime dt;              /* Date and Time */
         union
         {
            XmTkIwfSta     tkIwfSta;     /* Trunking IWF Status */
            XmPh2TkIwfSta  ph2TkIwfSta;  /* AAL1/AAL2 ph2 */
            XmVcciSta      vcciSta;      /* VCCI Status */
            XmCIDSta       cidSta;       /* CID Status */
            XmCicSta       cicSta;       /* Narrowband CIC Status */
            XmAtmProfSta   atmProfSta;   /* ATM profile Status */
            XmVtoaProfSta  vtoaProfSta;  /* VTOA profile Status */
            XmSigVcciSta   sigVcciSta;   /* AAL1/AAL2 SIGVCCI Status */
            XmPh2AtmRscSta ph2AtmRscSta; /* ph2 ATM Resource Status */
            SystemId       sysId;        /* System Id */
         }s;
      } ssta;

   } t;
} XmMngmt;
```

hdr

Header structure. For a description, refer to Section 3.3.1, "Header."

The type of status information desired can be selected by programming the header substructure as:

elmnt

Element. The allowable values are:

Name	Description
STXMFEATTRPIWF	Feature transparency IWF control block
STXMPH1TKIWF	Phase 1 trunking IWF control block
STXMPH2TKIWF	Phase 2 trunking IWF control block
STXMPH2RSCCB	Phase 2 ATM resource control block
STXMSIGVCCI	SIGVCCI control block
STXMVCCI	VCCI control block
STXMCID	CID control block
STXMCIC	CIC control block
STXMATMPROF	ATM profile configuration
STXMVTOAPROF	VTOA profile configuration

cfm

It is not valid in the status request.

dt

Date and time structure.

tkIwfSta

It is used for obtaining status for a phase 1 AAL trunking/feature transparency IWF control block. The **intfc** field specifies the IWF for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For a description, refer to Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmTkIwfSta
{
   RmInterface intfc;         /* trunking interface */
   Cntr        numActvCalls;  /* Calls Active on this IWF */
   U8          state;         /* State of trunking IWF CB */
} XmTkIwfSta;
```

ph2TkIwfSta

It is used for obtaining the status for a phase 2 AAL trunking IWF control block. The **intfc** field specifies the IWF for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For more details, refer to Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmPh2TkIwfSta   /* phase 2 trunking IWF CB */
{
   RmInterface intfc;         /* trunking interface */
   Cntr        numActvCalls;  /* Calls Active on this IWF */
   U8          state;         /* State of ph 2 trunking IWF CB */
} XmPh2TkIwfSta;
```

vcciSta

VCCI status structure. The **intfc** and **vcci** fields specify the VCCI for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For more details, refer to Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmVcciSta
{
   RmInterface  intfc;        /* trunking interface */
   Vcci   vcci;               /* VCCI */
   U8     state;              /* State of VCCI */
   U8     numActvCalls;       /* Calls Active on this VCCI */
} XmVcciSta;
```

cidSta

CID status structure. The **intfc**, **vcci**, and **cid** fields specify the CID for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For more details, refer to Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct XmCIDSta
{
   RmInterface  intfc;        /* trunking interface */
   Vcci   vcci;               /* VCCI */
   CID    cid;                /* CID */
   U8     state;              /* State */
   UConnId suConnId;          /* User Holding the Resource */
} XmCIDSta;
```

cicSta

Circuit status structure. The **intfc** and **cic** fields specify the circuit for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For details, see Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmCicSta
{
   RmInterface  intfc;       /* trunking interface */
   U16    cic;               /* Circuit Identification Code */
   U8     state;             /* State of CIC */
   UConnId suConnId;         /* User Holding the Resource */
} XmCicSta;
```

atmProfSta

It is used for obtaining status for a configured ATM profile control block. The **profId** field specifies the ATM profile for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For a description, refer to Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmAtmProfSta
{
   U8   profId;              /* Profile Identifier */
   Cntr numVccis;            /* number of VCCIs using this profile */
} XmAtmProfSta;
```

vtoaProfSta

It is used for obtaining status for a configured VTOA profile control block. The **profId** field specifies the VTOA profile for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For more description, see Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmVtoaProfSta
{
   U8   profId;              /* Profile Identifier */
   Cntr numCIDs;             /* number of CIDs using this profile */
} XmVtoaProfSta;
```

sigVcciSta

It is used for obtaining status for a signalling VCCI control block. The **intfc** and **vcci** fields specify the signalling VCCI for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For more description, refer to Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmSigVcciSta
{
   RmInterface intfc;        /* trunking interface */
   Vcci        vcci;         /* VCCI */
   U8          state;        /* State of VCCI */
} XmSigVcciSta;
```

ph2AtmRscSta

It is used for obtaining the status for a phase 2 ATM resource control block. The **intfc** field specifies the phase 2 ATM resource control block for which status information is requested. The other fields are not relevant in the status request. They are described in detail in the status confirm primitive. For details, see Section 3.7.1.2.8, "**RmMiLrmStaCfm**."

```
typedef struct xmPh2AtmRscSta  /* phase 2 ATM Resource CB */
{
   RmInterface intfc;         /* trunking interface */
   Cntr        numActvCalls;  /* Calls Active on this ATM resource CB */
   U8          state;         /* State of this ATM resource CB */
} XmPh2AtmRscSta;
```

Description:

The layer manager uses this function to gather solicited status information.

Returns:

00 ROK

01 RFAILED

3.9.1.2.8 XmMiLxmStaCfm

Name:

Status Confirm

Direction:

XM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 XmMiLxmStaCfm(pst, sta)
Pst     *pst;
RmMngmt *sta;
```

Parameters:

pst

For a description, see Section 3.3.3, "**Pst**."

sta

Pointer to the solicited status structure. The solicited status structure has the following format.

```
typedef struct xmMngmt
{
   Header hdr;
   CmStatus  cfm;                  /* status in confirm */
   union
   {
/* Solicited Status */
     struct
      {
         DateTime dt;              /* Date and Time */
         union
         {
            XmTkIwfSta     tkIwfSta;     /* Trunking IWF Status */
            XmPh2TkIwfSta  ph2TkIwfSta;  /* AAL1/AAL2 ph2 */
            XmVcciSta      vcciSta;      /* VCCI Status */
            XmCIDSta       cidSta;       /* CID Status */
            XmCicSta       cicSta;       /* Narrowband CIC Status */
            XmAtmProfSta   atmProfSta;   /* ATM profile Status */
            XmVtoaProfSta  vtoaProfSta;  /* VTOA profile Status */
            XmSigVcciSta   sigVcciSta;   /* AAL1/AAL2 SIGVCCI Status */
            XmPh2AtmRscSta ph2AtmRscSta; /* ph2 ATM Resource Status */
            SystemId       sysId;        /* System Id */
         }s;
      } ssta;
   } t;
} XmMngmt;
```

hdr

Header structure. For a description, refer to Section 3.3.1, "Header."

cfm

The status field indicates the result of a request. The status field has the following format:

```
typedef struct cmStatus
{
    U16   status;             /* Status of the operation  */
    U16   reason;             /* If failed, the reason    */
} CmStatus;
```

status

This field indicates the status of the previous control request primitive. It contains one of the following values.

Name	Description
`LCM_PRIM_OK`	Statistics request successful
`LCM_PRIM_NOK`	Statistics request failed

reason

In case of failure (**LCM_PRIM_NOK**), this field contains the cause of it. The following values are possible.

Name	Description
`LCM_REASON_INVALID_ENTITY`	Invalid entity passed in the header
`LCM_REASON_INVALID_INSTANCE`	Invalid instance passed in the header
`LCM_REASON_INVALID_MSGTYPE`	Invalid message type passed in the header
`LCM_REASON_GENCFG_NOT_DONE`	General configuration must be done before a control request can be processed
`LCM_REASON_INVALID_PAR_VAL`	Passed parameters are invalid. The specified resource is not configured

dt

Date and time structure.

tkIwfSta

It is used for obtaining the status for a phase 1 AAL trunking / feature transparency IWF control block.

```
typedef struct xmTkIwfSta
{
   RmInterface intfc;         /* trunking interface */
   Cntr        numActvCalls;  /* Calls Active on this IWF */
   U8          state;         /* State of trunking IWF CB */
} XmTkIwfSta;
```

intfc

The interface to which the feature transparency IWF control block belongs. This field must be filled when sending the status request.

numActvCalls

Number of active calls at the specified interface.

state

Reserved. This field is for future use. The state of the trunking IWF CB.

ph2TkIwfSta

It is used for obtaining the status for a phase 2 AAL trunking IWF control block.

```
typedef struct xmPh2TkIwfSta   /* phase 2 trunking IWF CB */
{
   RmInterface intfc;         /* trunking interface */
   Cntr        numActvCalls;  /* Calls Active on this IWF */
   U8          state;         /* State of ph 2 trunking IWF CB */
} XmPh2TkIwfSta;
```

intfc

The interface to which the feature transparency IWF control block belongs. This field must be filled when sending the status request.

numActvCalls

Number of active calls at the specified interface.

state

Reserved. This field is for future use. The state of the trunking IWF CB.

vcciSta

VCCI status structure.

```
typedef struct xmVcciSta
{
   RmInterface  intfc;       /* trunking interface */
   Vcci   vcci;              /* VCCI */
   U8     state;             /* State of VCCI */
   U8     actvCalls;         /* Calls Active on this VCCI */
} XmVcciSta;
```

intfc

Interface to which this VCCI belongs. This field must be filled when sending the status request.

vcci

AAL1 / AAL2 VCCI.

state

The state of the VCCI.

actvCalls

The number of calls currently active on this VCCI.

cidSta

The CID status structure is:

```
typedef struct XmCIDSta
{
   RmInterface  intfc;       /* trunking interface */
   Vcci   vcci;              /* VCCI */
   CID    cid;               /* CID */
   U8     state;             /* State */
   UConnId suConnId;         /* User Holding the Resource */
} XmCIDSta;
```

intfc

Interface to which this CID belongs. This field must be filled when sending the status request.

vcci

AAL1/AAL2 VCCI. This field must be filled when sending the status request.

cid

AAL1/AAL2/feature transparency CID on the given VCCI. This field must be filled when sending the status request.

state

The state of the CID.

suConnId

ID of the connection to which this CID is currently allocated.

cicSta

Circuit status structure.

```
typedef struct xmCicSta
{
   RmInterface  intfc;       /* trunking interface */
   U16    cic;               /* Circuit Identification Code */
   U8     state;             /* State of CIC */
   UConnId suConnId;         /* User Holding the Resource */
} XmCicSta;
```

intfc

Interface to which this circuit belongs. This field must be filled when sending the status request.

cic

Circuit ID. This field must be filled when sending the status request.

state

The state of the circuit. The state can have one of the following values.

Value	Description
LXM_IDLE	Resource is IDLE
LXM_UNEQUIP	Resource is unequipped
LXM_CP_BSY	Resource is CP busy
LXM_MNT_BSY	Resource is maintenance busy

suConnId

ID of the connection to which this CIC is currently allocated.

atmProfSta

It is used for obtaining status for a configured ATM profile control block.

```
typedef struct xmAtmProfSta
{
   U8   profId;              /* Profile Identifier */
   Cntr numVccis;            /* number of VCCIs using this profile */
} XmAtmProfSta;
```

profId

ATM profile ID. This field must be filled when sending the status request.

numVccis

Number of VCCIs currently using this ATM profile.

vtoaProfSta

It is used for obtaining status for a configured VTOA profile control block.

```
typedef struct xmVtoaProfSta
{
   U8   profId;              /* Profile Identifier */
   Cntr numCIDs;             /* number of CIDs using this profile */
} XmVtoaProfSta;
```

profId

VTOA profile ID. This field must be filled when sending the status request.

numCIDs

Number of CIDs currently using this VTOA profile.

sigVcciSta

It is used for obtaining status for a signalling VCCI control block.

```
typedef struct xmSigVcciSta
{
   RmInterface intfc;        /* trunking interface */
   Vcci        vcci;         /* VCCI */
   U8          state;        /* State of VCCI */
} XmSigVcciSta;
```

intfc

The interface for which the signalling VCCI control block is defined. This field must be filled when sending the status request.

vcci

The signalling VCCI ID. This field must be filled when sending the status request.

state

The state of the signalling VCC connection.

ph2AtmRscSta

It is used for obtaining the status for a phase 2 ATM resource control block.

```
typedef struct xmPh2AtmRscSta  /* phase 2 ATM Resource CB */
{
   RmInterface intfc;         /* trunking interface */
   Cntr        numActvCalls;  /* Calls Active on this ATM resource CB */
   U8          state;         /* State of this ATM resource CB */
} XmPh2AtmRscSta;
```

intfc

The interface to which the ATM resource belongs. This field must be filled when sending the status request.

numActvCalls

Number of active calls on the specified ATM resource.

state

Reserved. This field is for future use. State of the ATM resource.

Description:

The XM uses this function to return solicited status information to the layer manager.

Returns:

00 ROK

01 RFAILED

3.9.1.2.9 XmMiLxmStaInd

Name:

Status Indication

Direction:

XM to the layer manager

Supplied:

No

Synopsis:

```
PUBLIC S16 XmMiLxmStaInd(pst, sta)
Pst     *pst;
CcMngmt *sta;
```

Parameters:

pst

For a description, see Section 3.3.3, "**Pst**."

sta

Pointer to the unsolicited status structure. The status structure has the following format.

```
typedef struct xmMngmt
{
   Header    hdr;
   CmStatus  cfm;
   union
   {
/* unsolicited status */
      struct
      {
         CmAlarm alarm;            /* Alarm */
         XmDiag  diagn;            /* Diagnostics (if any) */
      } usta;
   } t;
} XmMngmt;
```

hdr

Header structure. For details, refer to Section 3.3.1, "Header."

cfm

The status field has no significance in this primitive.

alarm

Alarm. It contains the category, event, and cause of the alarm. These fields are described next.

```
typedef struct cmAlarm
{
   DateTime dt;       /* data and time */
   U16 category;      /* alarm category*/
   U16 event;         /* alarm event */
   U16 cause;         /* alarm cause */
}CmAlarm;
```

dt

Date and time structure.

category

This field describes the category to which the error belongs.

Name	Description
LCM_CATEGORY_PROTOCOL	Protocol error. This can occur while mapping one message from the incoming to the outgoing side.
LCM_CATEGORY_RESOURCE	The XM cannot allocate memory.
LCM_CATEGORY_INTERFACE	When an event is received on an SAP that is not configured or bound
LCM_CATEGORY_INTERNAL	Internal errors, such as hash list failures

event

This field specifies the event that has occurred.

Name	Description
LCM_EVENT_INV_STATE	Event received in an invalid state
LCM_EVENT_LYR_SPECIFIC	Protocol layer-specific mapping error
LXM_EVENT_HASHING_FAILED	Hash list operation failure
LXM_EVENT_MEMALOC_FAILED	Memory allocation failure

Name	Description
LXM_EVENT_AAL1_SIGSVC	AAL1 signalling VCC-related event
LXM_EVENT_AAL2_SIGSVC	AAL2 signalling VCC-related event
LXM_EVENT_AAL1_BEARERSVC	AAL1 bearer VCC-related event
LXM_EVENT_AAL2_BEARERSVC	AAL2 bearer VCC-related event
LXM_EVENT_FEATTRP_BEARERSVC	Feature transparency bearer VCC-related event

cause

This field specifies the cause. The additional information in union **t** depends on the cause.

Name	Description
LCM_CAUSE_INV_SAP	Invalid SAP. The value causing the problem is passed in the **suId** field.
LXM_CAUSE_INV_CID	The call control ID for a GCC primitive is out-of-range.
LXM_CAUSE_MALLOC_FAIL	The XM could not allocate memory.
LXM_CAUSE_TKINTFCTBL_FIND	The XM could not find the IWF block in the trunking IWF block hash list.
LXM_CAUSE_TKINTFCTBL_INS	The XM could not insert the IWF block in the trunking IWF block hash list.
LXM_CAUSE_SPCONNTBL_FIND	The XM could not find the connection in the trunking **spConnId** hash list.
LXM_CAUSE_SPCONNTBL_INS	The XM could not insert the connection in the trunking **spConnId** hash list.
LXM_CAUSE_VCCITBL_FIND	The XM could not find the VCCI block in the trunking VCCI block hash list.
LXM_CAUSE_VCCITBL_INS	The XM could not insert the VCCI block in the trunking VCCI block hash list.
LXM_CAUSE_SIGVCCESTABLISHED	The XM successfully established a signalling SVC.
LXM_CAUSE_BEARERVCCESTABLISHED	The XM successfully established a bearer SVC.
LXM_CAUSE_SIGVCCRELEASED	The XM released a signalling SVC.
LXM_CAUSE_BEARERVCCRELEASED	The XM released a bearer SVC.

Name	Description
LXM_CAUSE_WRONGATMTFCDESCUSED	The ATM traffic descriptor used for the SVC connection differs from the required default.
LXM_CAUSE_WRONGAALPARAMUSED	The AAL parameters used for the SVC connection differs from the required default.
LXM_CAUSE_WRONGBBEARCAPUSED	The broadband bearer capability used for the SVC connection differs from the required default.
LXM_CAUSE_WRONGQOSPARAMUSED	The ATM QoS parameters used for the SVC connection differs from the required default.

diagn

Diagnostics. It identifies the parameters associated with the alarm / event.

```
typedef struct xmDiag
{
   union
   {
      XmSVCInfo svcInfo;
    }s;
}XmDiag;
```

svcInfo

This field is filled when the alarm cause indicates an **LXM_CAUSE_SIGVCC_ESTABLISHED** or **LXM_CAUSE_BEARERVCC_ESTABLISHED** alarm. This field contains detailed information about the established SVC.

```
typedef struct xmSVCInfo
{
   U8 vcciType;       /* type of VCCI established */
   Vcci vcci;         /* VCCI value */
   Bool isItCntrlg    /* Is the VCCI controlling or controlled */
   RmInterface intfc; /* the trunking interface to which */
                      /* the given VCCI belongs */
   AalConParam aalConnParam; /* the AAL connection parameters */
                             /* for the VCCI */
  }XmSVCInfo;
```

vcciType

Flag to indicate the type of established AAL1 / AAL2 VCCI.

Value	Description
LXM_SIGVCCI_OVERAAL5	Signalling VCCI over AAL5
LXM_SIGVCCI_OVERAAL2	Signalling VCCI over AAL2
LXM_BEARERVCCI_AAL2	AAL2 bearer VCCI
LXM_BEARERVCCI_AAL1_SINGLE	One-to-one AAL1 bearer VCCI
LXM_BEARERVCCI_AAL1_MUX	Many-to-one AAL1 bearer VCCI
LXM_VCCI_FEATTRP	Feature transparency VCCI

vcci

Vcci of this resource for the AAL1 / AAL2 / feature transparency trunking interface.

IsItCntrlg

Indicates whether the VCCI is a controlling or controlled VCCI.

intfc

The ISDN interface to which the given VCCI belongs. For more description, see Section 3.3.6, "Interface."

aalConParam

The AAL connection parameters are used for establishing the given SVC connection. This information is valid only if the **vcciType** indicates a signalling VCCI (for example, **LXM_SIGVCCI_OVERAAL5** or **LXM_SIGVCCI_OVERAAL2**).

Description:

The XM uses this function to provide the layer manager with unsolicited status information (alarms). The unsolicited status can be enabled or disabled via the layer manager control request. The following table contains information on the alarms generated by the XM.

Description	Category	Event	Cause
An AAL1 signalling SVC was successfully established.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL1_SIGSVC**	**LXM_CAUSE_SIGVCCESTABLISHED**
An AAL1 bearer SVC was successfully established.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL1_BEARERSVC**	**LXM_CAUSE_BEARERVCCESTABLISHED**
An AAL1 signalling SVC was successfully released.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL1_SIGSVC**	**LXM_CAUSE_SIGVCCRELEASED**
An AAL1 bearer SVC was successfully released.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL1_BEARERSVC**	**LXM_CAUSE_BEARERVCCRELEASED**
An AAL2 signalling SVC was successfully established.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL2_SIGSVC**	**LXM_CAUSE_SIGVCCESTABLISHED**
An AAL2 bearer SVC was successfully established.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL2_BEARERSVC**	**LXM_CAUSE_BEARERVCCESTABLISHED**
An AAL2 signalling SVC was successfully released.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL2_SIGSVC**	**LXM_CAUSE_SIGVCCRELEASED**
An AAL2 bearer SVC was successfully released.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_AAL2_BEARERSVC**	**LXM_CAUSE_BEARERVCCRELEASED**
A feature transparency bearer SVC was successfully established.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_FEATTRP_BEARERSVC**	**LXM_CAUSE_BEARERVCCESTABLISHED**
A feature transparency bearer SVC was successfully released.	**LCM_CATEGORY_PROTOCOL**	**LXM_EVENT_FEATTRP_BEARERSVC**	**LXM_CAUSE_BEARERVCCRELEASED**

Returns:

00 ROK

01 RFAILED

3.9.2 Interface with the Upper Layers

The XM is the service provider for GCC. It also acts as the RM and service user, which initiates and terminates calls. It is a combined functionality of the PSIF and RM, therefore, the XM has two upper interfaces: one for resource management (RMT) and one for call establishment (CCT).

3.9.2.1 Resource Management Interface

The following primitives are provided at the interface between the XM and GCC for managing trunking resources. The interface is called the RMT interface.

Bind Establishment

Primitive Name	Description	Flow
`XxYyRmtBndReq`	Bind request	GCC to XM
`XxYyRmtBndCfm`	Bind confirm	XM to GCC

Resource Management Bind

Primitive Name	Description	Flow
`XxYyRmtAlocReq`	Resource allocation request	GCC to XM
`XxYyRmtAlocCfm`	Resource allocation confirm	XM to GCC
`XxYyRmtDalocReq`	Resource deallocation request	GCC to XM
`XxYyRmtDalocCfm`	Resource deallocation confirm	XM to GCC
`XxYyRmtDalocInd`	Resource deallocation indication	XM to GCC

For a detailed description of the RMT interface, refer to the *Interworking Call Control Interface Service Definition*.

3.9.2.2 CCT Interface

The following tables list primitives used between GCC and the XM to initiate and terminate connections. For a detailed description, refer to the *CCT Interface Service Definition*.

Bind Establishment

Primitive Name	Description	Flow
`XxYyCctBndReq`	Bind request	GCC to XM
`XxYyCctBndCfm`	Bind confirm	XM to GCC

Generic Call Control

Primitive Name	Description	Flow
`XxYyCctConInd`	Connection establishment indication	XM to GCC
`XxYyCctConReq`	Connection establishment request	GCC to XM
`XxYyCctConCfm`	Connection establishment confirm	XM to GCC
`XxYyCctConRsp`	Connection establishment response	GCC to XM
`XxYyCctAddrInd`	Additional addressing indication	XM to GCC
`XxYyCctRscCfm`	Resource confirm	XM to GCC
`XxYyCctRscRsp`	Resource response	GCC to XM
`XxYyCctCnStInd`	Connection status indication	XM to GCC
`XxYyCctCnStReq`	Connection status request	GCC to XM
`XxYyCctRelInd`	Release indication	XM to GCC
`XxYyCctRelReq`	Release request	GCC to XM
`XxYyCctRelRsp`	Release response	GCC to XM
`XxYyCctRelCfm`	Release confirm	XM to GCC
`XxYyCctStaInd`	Status indication	XM to GCC
`XxYyCctModInd`	Modification indication	XM to GCC
`XxYyCctModReq`	Modification request	GCC to XM
`XxYyCctModRsp`	Modification response	GCC to XM
`XxYyCctModCfm`	Modification confirm	XM to GCC
`XxYyCctHldInd`	Connection hold indication	XM to GCC
`XxYyCctRtrInd`	Connection retrieve indication	XM to GCC
`XxYyCctProfInd`	Profile indication	XM to GCC

3.9.3 Interface with System Services

This section discusses the XM's interface with system services.

3.9.3.1 General

This section describes the system services required by the XM.

Task Scheduling

The task scheduling management functions are called to register, activate, and terminate a task. Use the following functions for task scheduling management.

Name	Description
`SRegActvTsk`	Register activate task - Task
`xmActvTsk`	Activate the task for the XM
`SPstTsk`	Post task
`SExitTsk`	Exit task

Initialization

OS calls the initialization management function to initialize a task. Use the following function for initialization management.

Name	Description
`xmActvInit`	Activate task - Initialize the XM

Memory Management

Using static buffers, the memory management functions allocate and deallocate variable-sized buffers. Use the following functions for memory management.

Name	Description
`SGetSBuf`	Get static buffer
`SGetSMem`	Get static memory

Message Management

The message management functions initialize, add data to, and remove data from messages utilizing dynamic buffers. Use the following functions for message management.

Name	Description
`SGetMsg`	Allocate a message (from a dynamic pool)
`SPutMsg`	Deallocate a message (into a dynamic pool)
`SFndLenMsg`	Find the length of a message
`SAddPreMsg`	Add an octet to the beginning of a message
`SAddPstMsg`	Add an octet to the end of a message
`SRemPreMsg`	Remove an octet from the beginning of a message
`SRemPstMsg`	Remove an octet from the end of a message
`SPkS8`	Add a signed 8-bit value to a message
`SPkU8`	Add an unsigned 8-bit value to a message
`SPkS16`	Add a signed 16-bit value to a message
`SPkU16`	Add an unsigned 16-bit value to a message
`SPkS32`	Add a signed 32-bit value to a message
`SPkU32`	Add an unsigned 32-bit value to a message
`SUnpkS8`	Remove a signed 8-bit value from a message
`SUnpkU8`	Remove an unsigned 8-bit value from a message
`SUnpkS16`	Remove a signed 16-bit value from a message
`SUnpkU16`	Remove an unsigned 16-bit value from a message
`SUnpkS32`	Remove a signed 32-bit value from a message
`SUnpkU32`	Remove an unsigned 32-bit value from a message

Timer Functions

Name	Description
`SRegTmr`	Registers the activation function - Timer
`SDeRegTmr`	Deregisters the activation function - Timer
`xmPrcIwfTq`	Timer activation function for the XM. It is used to process internal timers.

Miscellaneous

Resource availability checking. The following miscellaneous functions are used.

Name	**Description**
SFndProcId	Find processor ID on which a task runs
SGetDateTime	Get real date and time
SLogError	Handle an error
SPrint	Print a preformatted string to the default display device

For a detailed description of the system services listed above, refer to the *System Services Interface Service Definition*.

3.9.3.2 xmActvInit

Name:

Activate Task - Initialize the XM

Direction:

System services to the XM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 xmActvInit(ent, inst, region, reason)
Ent    ent;
Inst   inst;
Region region;
Reason reason;
```

Parameters:

`ent`

Entity ID.

`inst`

Instance ID for the entity.

`region`

Memory region ID that may be used by the layer to get static memory.

`reason`

Reason for initialization. Currently, this field is not used.

Description:

System services uses this function to initialize the XM.

Returns:

`00      ROK`

`01      RFAILED`

3.9.3.3 xmActvTsk

Name:

Activate Task

Direction:

System services to the XM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 xmActvTsk(pst, mBuf)
Pst    *pst;
Buffer *mBuf;
```

Parameters:

`pst`

Destination post structure. For a description, refer to Section 3.3.3, "**`Pst`**."

`mBuf`

Message buffer.

Description:

System services uses this function, which injects an event and a primitive into the XM. The given message buffer is unpacked to find the corresponding primitive and associated parameters. Then, the appropriate primitive reception handler is scheduled.

Returns:

`00      ROK`

`01      RFAILED`

3.9.3.4 xmPrcIwfTq

Name:

Activate Task - Timer

Direction:

System services to the XM

Supplied:

Yes

Synopsis:

```
PUBLIC S16 xmPrcIwfTq()
```

Parameters:

None

Description:

System services uses this function to activate timers with a timer tick. While it processes the general configuration request, the XM registers this function with system services. The XM uses the **SRegTmr** system services primitive and passes the pointer to **xmPrcIwfTq** as an argument to register the XM timer function with system services. The period during which this timer function must be invoked is also passed in **SRegTmr**.

Returns:

00	**ROK**
01	**RFAILED**

4 INTERFACE PROCEDURES

This section describes the interface procedures defined for ICC.

The interface procedures define the mechanisms by which ICC interacts, via primitives, with any adjacent software in the system in which it resides.

For each flow diagram in this section, the following rules apply:

1. Time flows toward the bottom of the page.
2. The mnemonic above a line represents a function call or primitive.
3. The mnemonic below a line represents an event type.
4. The **+** indicates an OR condition—one path or another may be taken.
5. The **o** indicates an AND condition—both paths are taken in parallel.
6. The labels above each flow diagram have the following meaning.

Name	Description
`ss`	System services
`gcc`	Generic call control
`rt`	Router
`rm`	Resource manager
`sfm`	Switching fabric manager
`xm`	Connection manager
`ll`	Lower layer
`lm`	Layer manager

The interface procedures differ, depending on whether a tightly coupled or loosely coupled interface is used. The interface between the lower layers and the layer manager may be tightly coupled or loosely coupled.

Note: *The system services interface is always tightly coupled.*

A tightly coupled interface means that the interface between two protocol layers consists of direct function calls between the two layers.

A loosely coupled interface means that the interface between two protocol layers consists of passing messages between the two layers, via queues, maintained by system services.

If a tightly coupled interface is used, the primitives referenced in the flow diagrams are called directly.

The flow and message format for the steps applicable to the tightly coupled interface is:

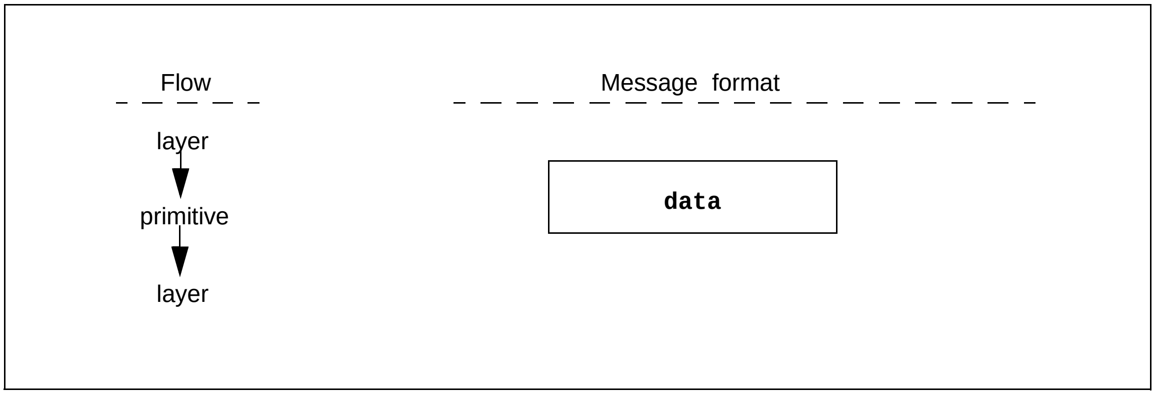


Figure 4-1Flow and message format, tightly coupled interface

If applicable, the data is the message buffer passed in the primitive. If a loosely coupled interface is used, a set of packing (**SPkxxx**) and unpacking (**SUnpkxxx**) functions sits between the primitive and the associated message, to and / or from system services.

The flow and message format for the steps applicable to the loosely coupled interface is:

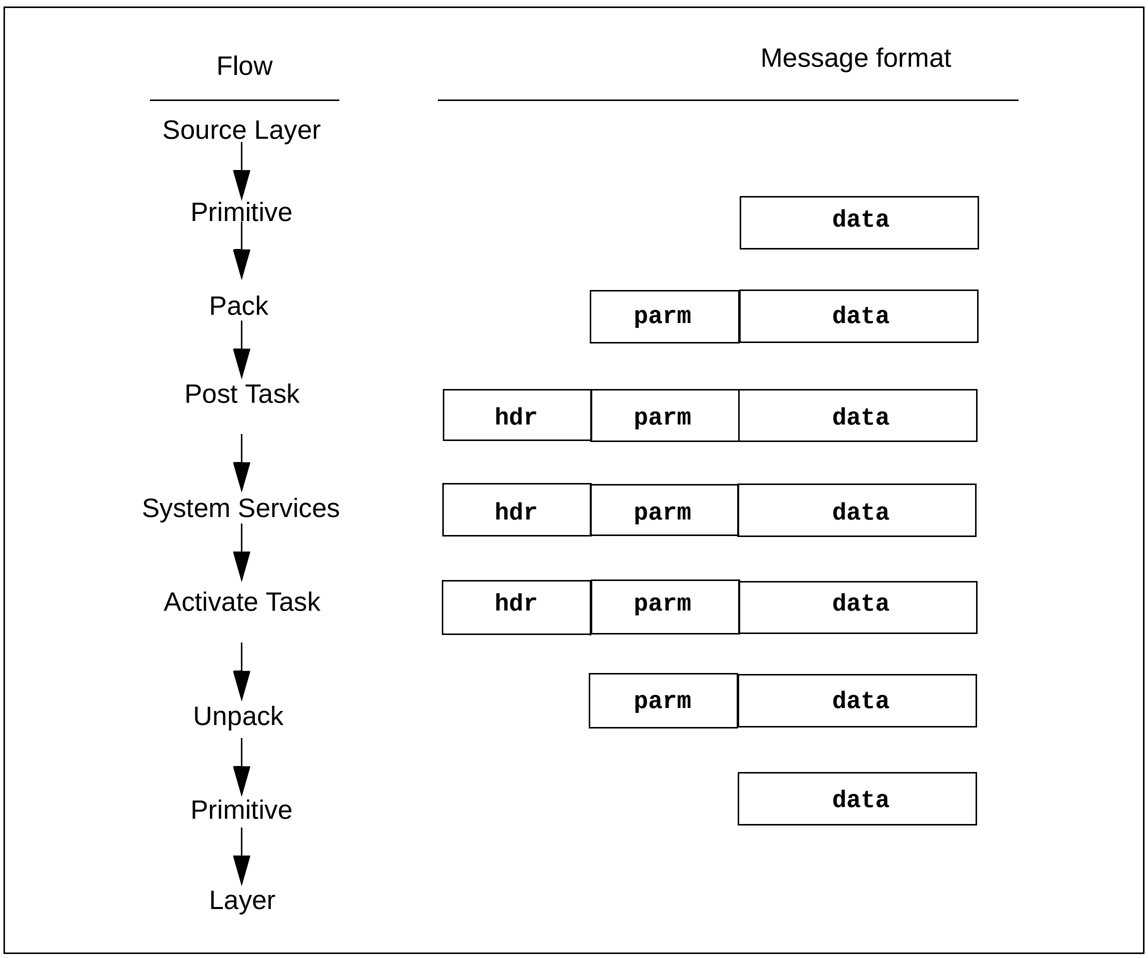


Figure 4-2Flow and message format, loosely coupled interface

The header (**`hdr`**) is independent of the protocol layer and represents information that system services must place in the message (priority, routing, destination entity, source entity) to ensure proper message routing to the destination entity.

The parameters (**`parm`**) are an encoded version of all the parameters passed in the primitive. If applicable, the **`data`** is the message buffer passed in the primitive.

A primitive causing a message to flow to another layer calls the appropriate packing function, then it calls the post task (**`SPstTsk`**) system services primitive to send the message to the layer. A message received from a layer causes system services to call the activate task (**`ccActvTsk`**) primitive, which calls the appropriate unpacking function, then it calls the primitive.

For clarity, the packing, unpacking, **`SPstTsk`**, and **`ccActvTsk`** primitives are not included in the flow diagrams for a loosely coupled interface.

4.1 Interface

The interface procedures define the mechanisms by which ICC interacts, via primitives, with any adjacent software in the system in which it resides.

The following interface procedures are supported:

- Initialization

  Procedures relating to initializing GCC, RT, RM, and SFM.

- Management

  Procedures relating to controlling and monitoring ICC (configuration, statistics, solicited status, unsolicited status, accounting, and control).

- Bind

  Procedures relating to binding ICC with the service user (upper layer) and service provider (network layer).

For proper operation, the interface procedures must be performed in the following order:

1. Initialization

   Initializes GCC, RT, RM, XM, and SFM (in any order).

2. Management - Configuration

   Configures GCC, RT, RM, XM, and SFM (in any order). Each entity has a specified configuration order, which is listed later.

3. Bind

   Bind GCC to its lower layers. This is performed via the control request stack start.

4. Management - Control (if required)

   Enables unsolicited status, if required. Unsolicited status can be enabled earlier, after general configuration is complete.

5. Management - Control

   Enables, disables, or deletes DPC. The possible operations depend on the entities. Refer to the different control requests for further explanation.

Following the stack start control request, ICC is ready for call processing.

The interface procedures are described in more detail in the following section.

4.1.1 Initialization

System services initiates the procedure that initializes ICC. The initialization primitive of all the entities (GCC, RT, RM, and SFM) must be called only once before any other primitives or functions are called. There is no fixed order in which the initialization primitives for all the entities are called.

Following initialization, ICC is ready for the management - configuration procedure.

The system services primitives are not called during the initialization procedure.

The data flow is:

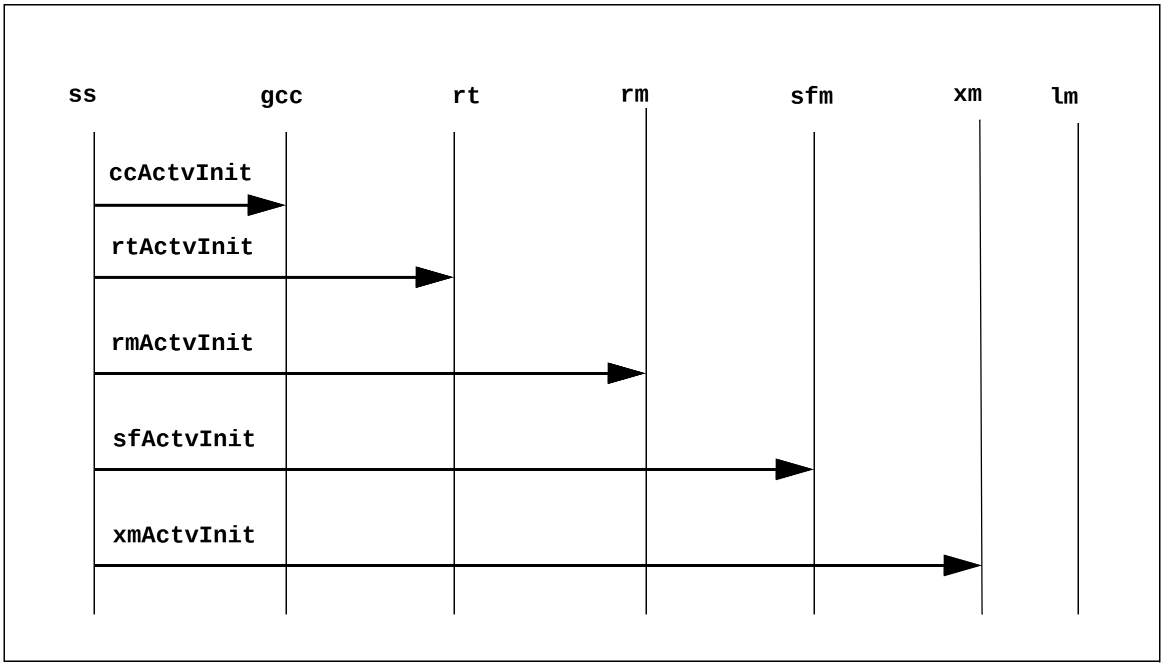


Figure 4-3Data flow: Initialization procedure

4.1.2 Management - Configuration

The layer manager initiates the management - configuration procedure to configure the various elements of ICC.

4.1.2.1 Generic Call Control Configuration

The layer manager initiates the management - configuration procedure to configure the various parts of GCC. The configuration request primitive (**`CcMiLccCfgReq`**) can be called more than once. The GCC configuration request primitives must be called before the bind primitives are called. Following the management - configuration procedure, GCC is ready for the bind procedure.

The following GCC configuration request primitive types may be called.

Name	**Description**
General	Passes parameters applying to GCC as a whole. It is used primarily to tune GCC for the most efficient use of its resources. It may be called only once.
PSIF SAP	Protocol-Specific Interface Function (PSIF) SAP configuration. It is used to communicate with incoming / outgoing protocols.
RM SAP	Configures the SAP toward the RM. It may be called only once.
RT SAP	Configures the SAP toward the RT. It may be called only once.
SFM SAP	Configures the SAP toward the SFM. It may be called only once.
Interface	It configures the GCC interface SAP table containing SAPs associated with an interface.
Virtual interface	Trunking / Trunked interface. It configures the GCC interface SAP table containing SAPs associated with an interface.
Profile	ATM parameters profile configuration. It contains information required for using CBR and VBR services, and for introducing in B-ISUP messages, certain parameters that have no correspondent in the original ISUP messages (for example, broadband bearer capability). It may be called one or more times. Configure this table before configuring narrowband circuits.
Observation table	It configures the table containing triggers for call trace analysis.

The **`CcMgmnt.hdr.elmntId`** field specifies the configuration request primitive type.

System services primitives are called during the management - configuration procedure. Upon completion of each successful or unsuccessful configuration, GCC sends a configuration confirmation primitive to the layer manager to notify the result of this operation.

For proper operation, the configuration request primitive type must be called in the following order:

1. General
2. PSIF / RM / RT / SFM SAPs
3. Interface / Virtual Interface
4. Profile
5. Observation table

The **CcMgmnt.t.cfg** structure specifies parameters used by the configuration request (**CcMiLccCfgReq**) primitive.

The data flow is:

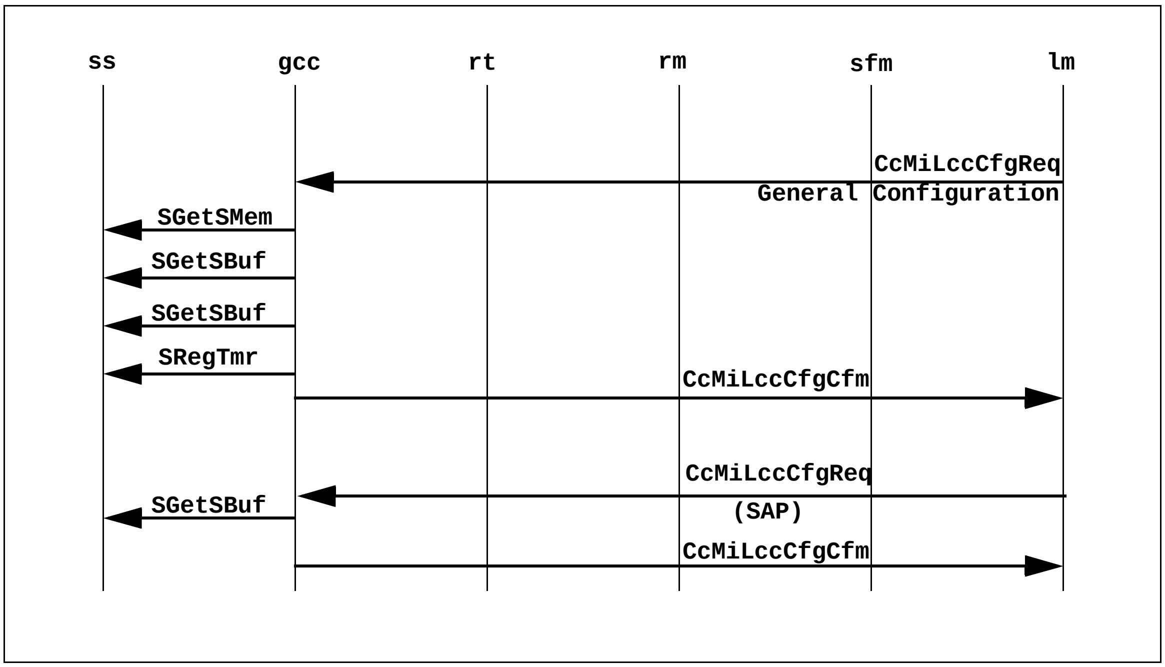


Figure 4-4Data flow: Management configuration procedure

4.1.2.2 Router Configuration

The layer manager initiates the management - configuration procedure to configure the RT. The configuration request primitive (**RtMiLrtCfgReq**) can be called more than once. The RT configuration request primitives must be called before GCC issues a bind to the RT. The RT software is ready for the bind procedure following the management - configuration procedure.

The following RT configuration request primitive types can be called.

Name	**Description**
General	Passes parameters applying to the whole RT. It is used primarily to calculate RT's memory requirements. It can be called only once.
Upper SAP	Used to communicate with GCC. It can be called one or more times.
Interface	Interface information to which routes can refer. It can be called one or more times.
Route	Route information. It can be called one or more times.
Virtual interface	Trunking / Trunked interface. Interface information to which routes can refer. It can be called one or more times.

The **RtMgmnt.hdr.elmntId** field specifies the configuration request primitive type.

The system services primitives are called during the management - configuration procedure. Upon completing each successful or unsuccessful configuration, the RT sends a configuration confirmation primitive to the layer manager to indicate the result of this operation.

For proper operation, the configuration request primitive types must be called in the following order:

1. General
2. Upper SAP
3. Interface / Virtual Interface
4. Route

The **RtMgmnt.t.cfg** structure specifies parameters used by the configuration request (**RtMiLrtCfgReq**) primitive.

The data flow is:

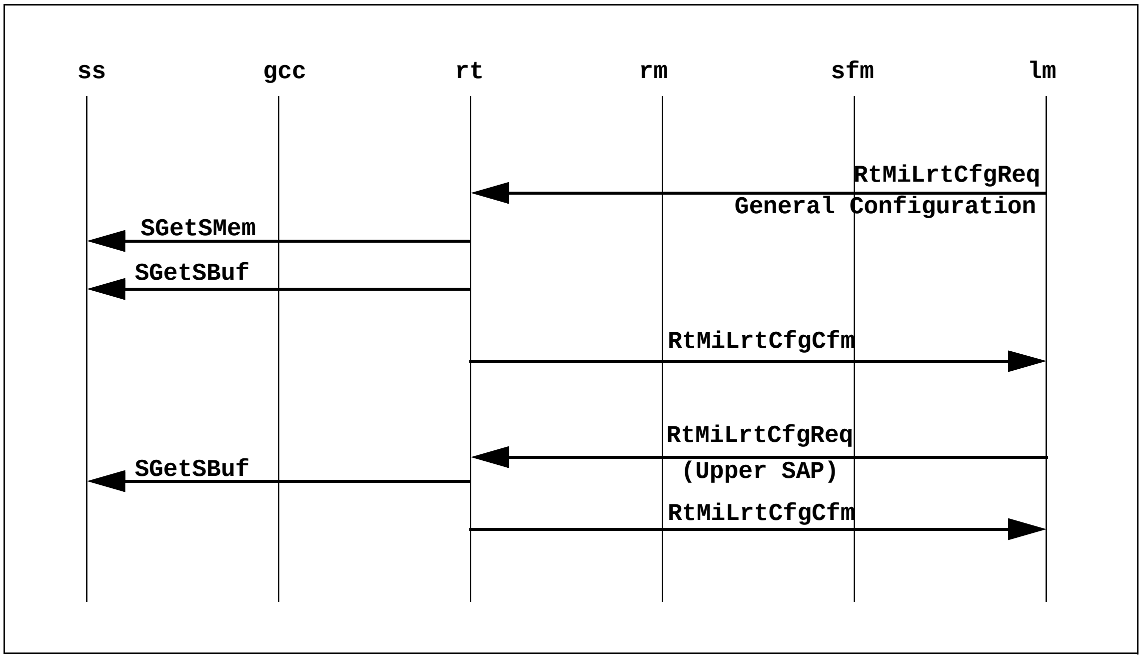


Figure 4-5Data flow: RT configuration

4.1.2.3 Resource Manager Configuration

The layer manager initiates the management - configuration procedure to configure the RM. The configuration request primitive (**`RmMiLrmCfgReq`**) can be called more than once. The RM configuration request primitives must be called before GCC or the PSIF issues a bind to the RM. The RM is ready for the bind procedure following the management - configuration procedure.

The following RM configuration request primitive types can be called.

Name	Description
General	Passes parameters applying to the whole RM. It is used primarily to calculate RM's memory requirements. It may be called only once.
Upper SAP	Used to communicate with GCC or the PSIF. It can be called one or more times.
Broadband interface	This is necessary only if ICC supports broadband interfaces (B-ISUP, Q.93B, and PNNI). It can be called one or more times.
VPI configuration	This is necessary only if ICC supports broadband interfaces (B-ISUP, Q.93B, and PNNI). It can be called one or more times.
Broadband physical link	This is necessary only if ICC supports broadband interfaces (B-ISUP, Q.93B, and PNNI). It can be called one or more times.
Narrowband DPC	This is necessary only if ICC supports ISUP. It can be called one or more times.
DSS1 interface	This is necessary only if ICC supports Q.930 / Q.931. It can be called one or more times.
Circuit	This is necessary only if ICC supports ISUP. It can be called one or more times.
Static binding	Binding between two resources (incoming and outgoing). This is necessary only if predefined binding is requested. In case of dynamic switching, such configurations must not be made. It can be called one or more times.

The **`RmMgmnt.hdr.elmntId`** field specifies the configuration request primitive type.

The system services primitives are called during the management - configuration procedure. Upon completing each successful or unsuccessful configuration, the RM sends a configuration confirmation primitive to the layer manager to indicate the result of this operation.

For proper operation, the configuration request primitive types must be called in the following order:

1. General
2. Upper SAP
3. Broadband physical link configuration
4. Interface (DPC configuration), narrowband DPC, broadband interface, and DSS1 interface
5. Resource configuration, VPI, and circuit configuration
6. Static binding configuration, if required

The **`RmMgmnt.t.cfg`** structure specifies parameters used by the configuration request (**`RmMiLrmCfgReq`**) primitive.

The data flow is:

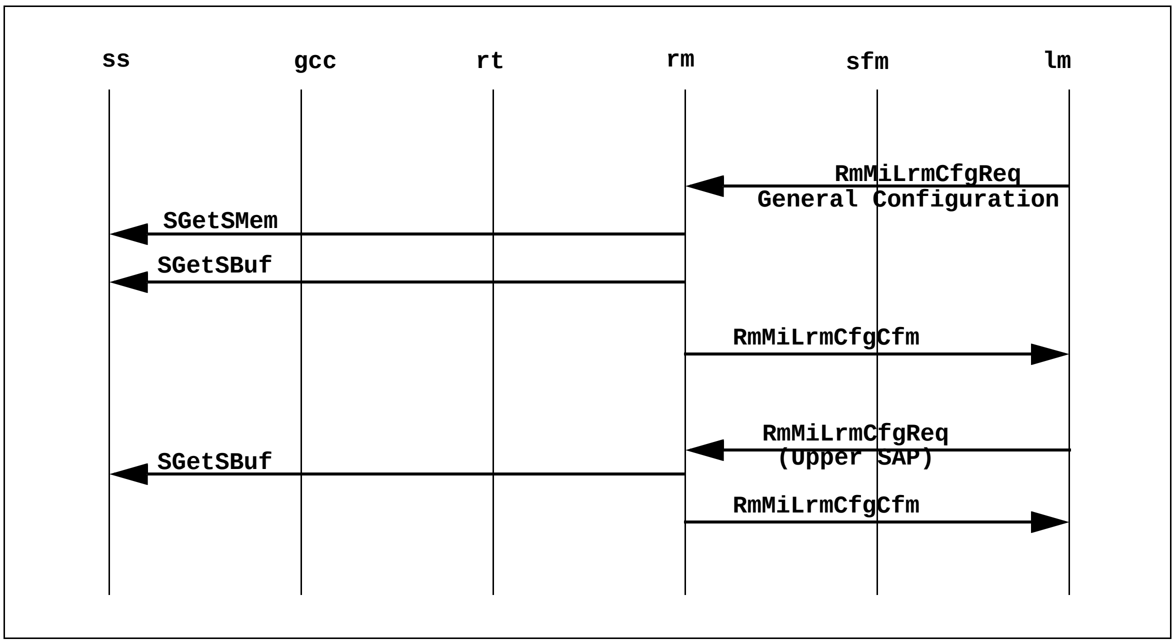


Figure 4-6Data flow: RM configuration

4.1.2.4 Switching Fabric Manager Configuration

The layer manager initiates the management - configuration procedure to configure the SFM. The configuration request primitive (**SfMiLsfCfgReq**) can be called more than once. The SFM configuration request primitives must be called before GCC issues a bind to the SFM. The SFM is ready for the bind procedure following the management - configuration procedure.

The following SFM configuration request primitive types can be called.

Name	**Description**
General	Passes parameters applying to the whole SFM. It is used primarily to calculate the memory requirements for the SFM. It can be called only once.
Upper SAP	Used to communicate with GCC. It can be called one or more times.

The **SfMgmnt.hdr.elmntId** field specifies the configuration request primitive type.

The system services primitives are called during the management - configuration procedure. Upon completing each successful or unsuccessful configuration, the SFM sends a configuration confirmation primitive to the layer manager to indicate the result of this operation.

For proper operation, the configuration request primitive types must be called in the following order:

1. General
2. Upper SAP

The **SfMgmnt.t.cfg** structure specifies parameters used by the configuration request (**SfMiLsfCfgReq**) primitive.

The data flow is:

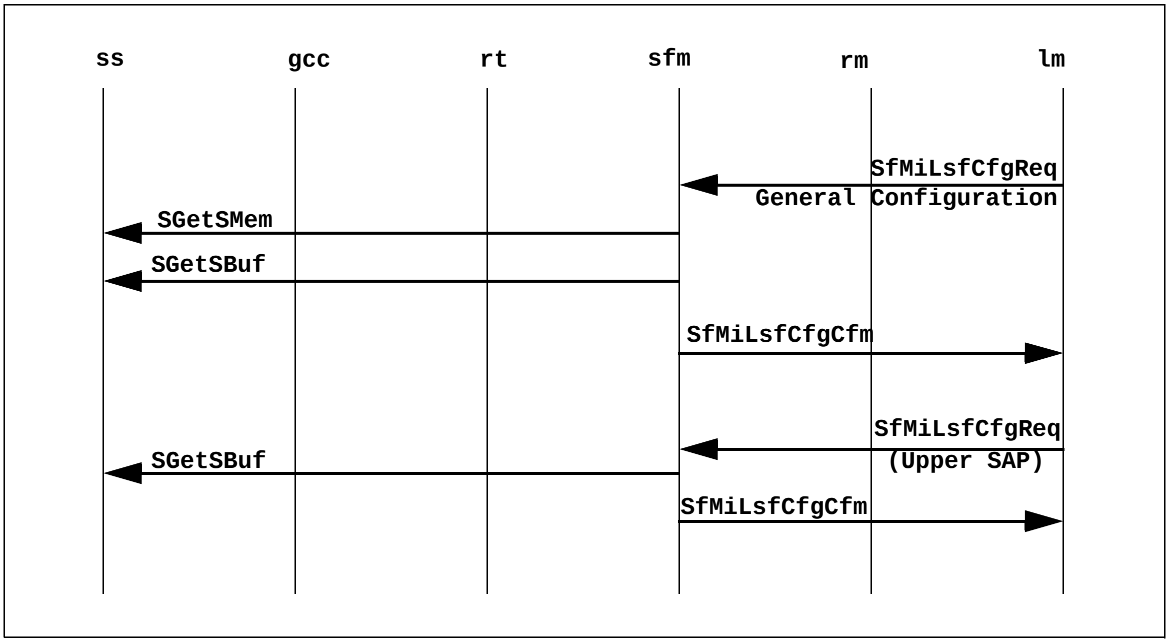


Figure 4-7Data flow: SFM configuration

4.1.2.5 Connection Manager Configuration

The layer manager initiates the management - configuration procedure to configure the XM. The configuration request primitive (**`XmMiLxmCfgReq`**) can be called more than once. The XM configuration request primitives must be called before GCC or the PSIF issues a bind to the XM. The XM is ready for the bind procedure following the management - configuration procedure.

The following XM configuration request primitive types can be called.

Name	**Description**
General	Passes parameters applying to the whole XM. It is used primarily to calculate the memory requirements. It can be called only once.
Upper CC SAP	It is used to communicate with GCC for terminating and initiating connections.
Upper RM SAP	It is used to communicate with GCC to manage the resources of the virtual interfaces.
Upper RM SAP	It is used to communicate with GCC to manage the resources of the virtual interfaces.
Trunking IWF control block	It is used to configure the parameters required for a phase 1 AAL1 / AAL2 or a feature transparency trunking IWF control block.
Phase 2 trunking IWF control block	It is used to configure the parameters required for a phase 2 AAL1 / AAL2 trunking IWF control block.
Signalling Virtual Channel Connection ID (VCCI) control block	It is used to configure the parameters required for the signalling VCCI control block of a phase 1 AAL1 / AAL2 trunking IWF control block.
VCCI control block	It is used to configure the parameters required for PVC-based VCCIs of a phase 1 AAL / AAL2 trunking IWF control block.
CIC control block	It is used to configure the parameters required for a CIC belonging to a phase 2 AAL1 / AAL2 trunking IWF control block.
Phase 2 ATM resource control block	It is used to configure the parameters required for a phase 2 AAL1 / AAL2 trunking resource control block.
ATM profile	It is used to configure parameters required for an ATM profile control block.
VTOA profile	It is used to configure parameters required for a VTOA profile control block.

The **XmMgmnt.hdr.elmntId** field specifies the configuration request primitive type.

The system services primitives are called during the management - configuration procedure. Upon completing each successful or unsuccessful configuration, the XM sends a configuration confirmation primitive to the layer manager to indicate the result of this operation.

For proper operation, the configuration request primitive types must be called in the following order:

1. General
2. CC Upper SAPs / RM Upper SAPs
3. ATM profiles, if required
4. VTOA profiles, if required
5. Trunking IWF control blocks or phase 2 trunking IWF control blocks
6. ATM phase 2 resource control blocks
7. Signalling VCCI control block
8. VCCI control blocks or CIC control blocks

The **XmMgmnt.t.cfg** structure specifies parameters used by the configuration request (**XmMiLxmCfgReq**) primitive.

The data flow is:

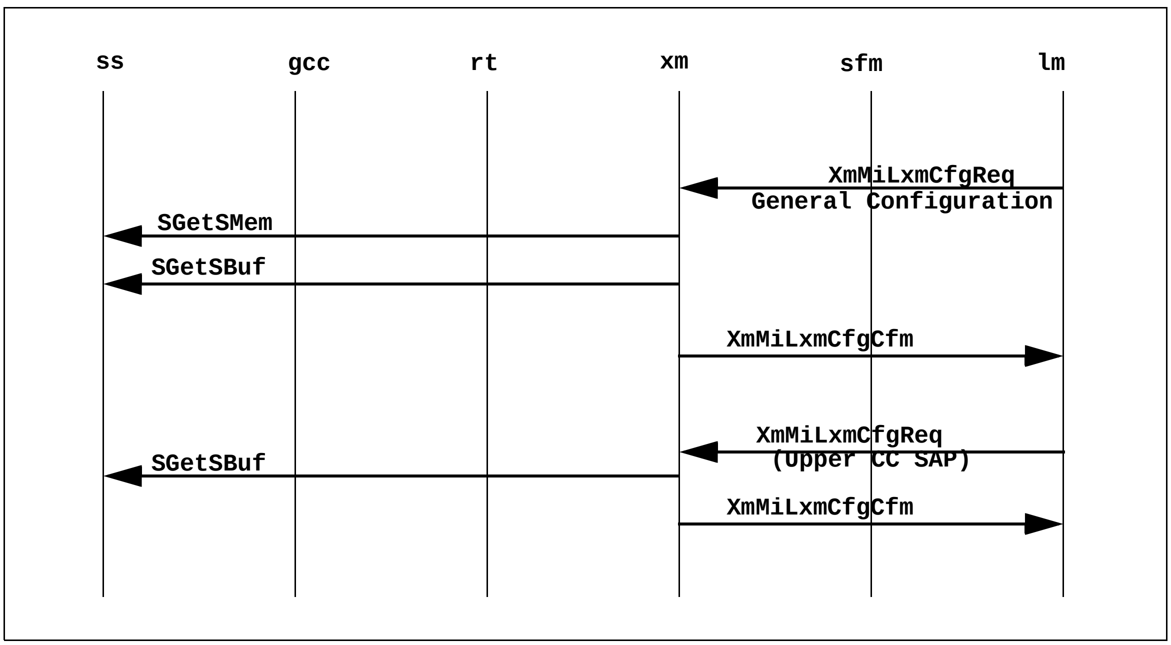


Figure 4-8Data flow: XM configuration

4.1.3 Management - Statistics

The layer manager uses the management - statistics procedure to gather statistics information about the various elements of ICC. Currently, the following entities provide statistics information:

- GCC
- RT
- RM
- XM

4.1.3.1 Generic Call Control

The layer manager initiates the management - statistics procedure to gather statistics information from GCC. GCC's statistics request primitive (**`CcMiLccStsReq`**) can be called more than once, any time after the management - configuration procedure.

The following statistics can be requested:

- General statistics
- Protocol (PSIF) SAP statistics
- Interface statistics

The **`ccMgmnt.hdr.elmId`** field specifies the statistics request primitive.

The statistics confirm (**`CcMiLccStsCfm`**) primitive is called during the statistics procedure after the statistics structure has been initialized.

The **`CcMgmnt.t.sts`** structure specifies parameters used by the statistics request and statistics confirm (**`CcMiLccStsReq`** and **`CcMiLccStsCfm`**) primitives.

The data flow is:

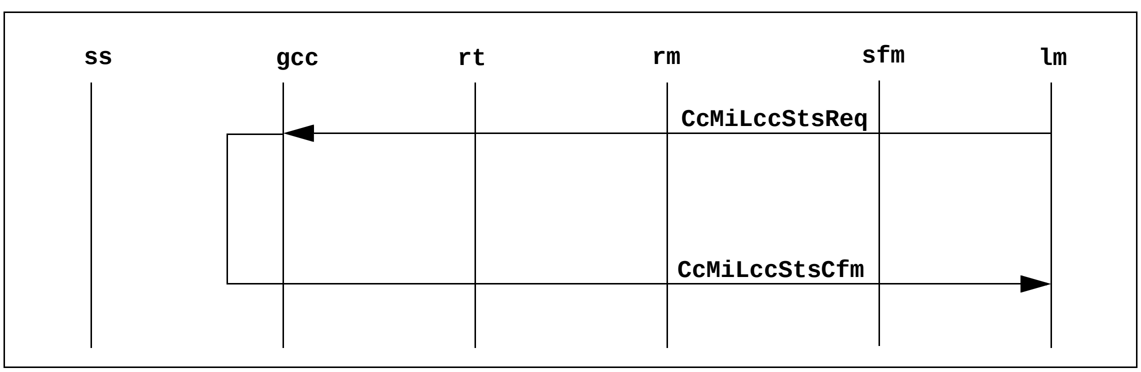


Figure 4-9Data flow: GCC statistics request procedure

4.1.3.2 Router

The layer manager initiates the management - statistics procedure and uses it to gather statistics information from the RT. The RT statistics request primitive (**RtMiLrtStsReq**) may be called more than once, any time after the management - configuration procedure.

The statistic per interface (DPC) can be requested.

The **rtMgmnt.hdr.elmId** field specifies the statistics request primitive type.

The statistics confirm (**RtMiLrtStsCfm**) primitive is called during the statistics procedure, after the statistic structure has been initialized.

The **RtMgmnt.t.sts** structure specifies parameters used by the statistics request and statistics confirm (**RtMiLrtStsReq** and **RtMiLrtStsCfm**) primitives.

The data flow is:

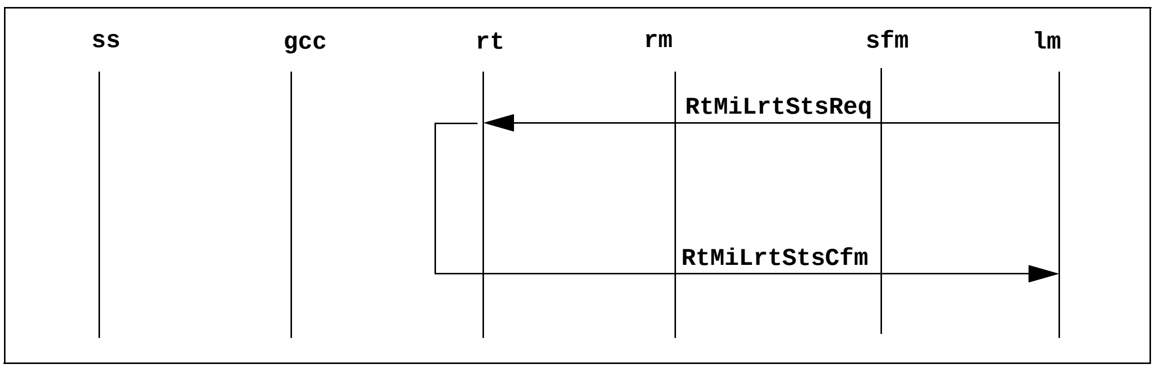


Figure 4-10Data flow: RT statistics request procedure

4.1.3.3 Resource Manager

The layer manager initiates the management - statistics procedure to gather statistics information from the RM. The RM statistics request primitive (**RmMiLrmStsReq**) can be called more than once, any time after the management - configuration procedure.

The following statistics can be requested.

- Statistics per narrowband interface (narrowband DPC)
- Statistics per broadband interface (broadband DPC)

The **rmMgmnt.hdr.elmId** field specifies the statistics request primitive type.

The statistics confirm (**RmMiLrmStsCfm**) primitive is called during the statistics procedure, after the statistic structure has been initialized.

The **RmMgmnt.t.sts** structure specifies the parameters used by the statistics request and statistics confirm (**RmMiLrmStsReq** and **RmMiLrmStsCfm**) primitives.

The data flow is:

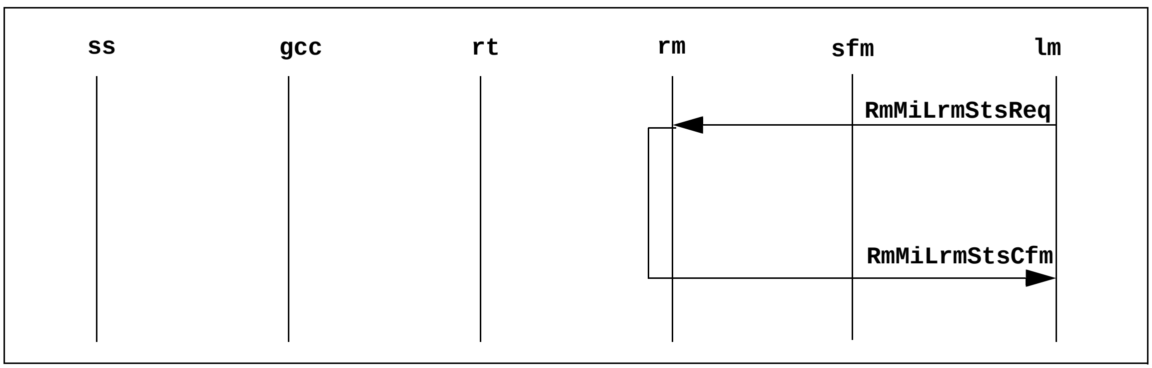


Figure 4-11Data flow: RM statistics request procedure

4.1.3.4 Connection Manager

The layer manager initiates the management - statistics procedure to gather statistics information from the XM. The XM statistics request primitive (**XmMiLxmStsReq**) can be called more than once, any time after the management - configuration procedure.

The statistics per IWF can be requested.

The **xmMgmnt.hdr.elmId** field specifies the statistics request primitive type.

The statistics confirm (**XmMiLxmStsCfm**) primitive is called during the statistics procedure, after the statistic structure has been initialized.

The **XmMgmnt.t.sts** structure specifies the parameters used by the statistics request and statistics confirm (**XmMiLxmStsReq** and **XmMiLxmStsCfm**) primitives.

The data flow is:

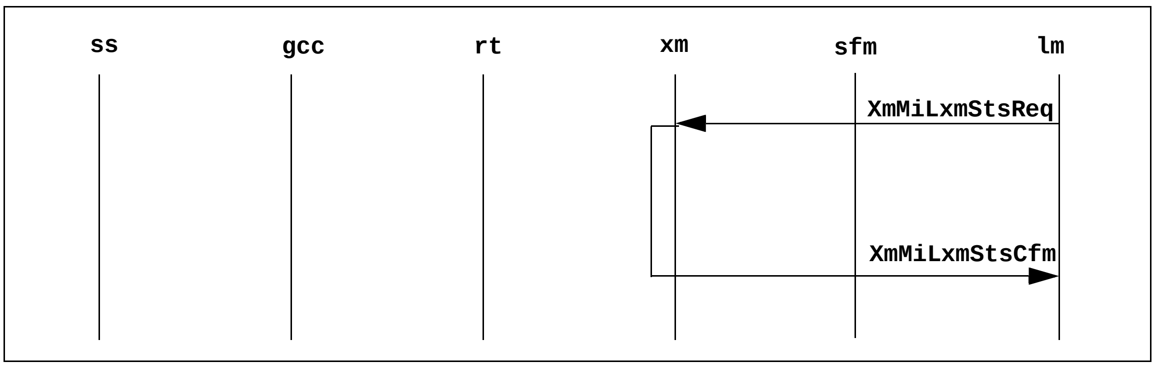


Figure 4-12Data flow: The XM statistics request procedure

4.1.4 Management - Solicited Status

The layer manager uses the management - solicited status procedure to gather solicited status from the various entities of ICC. Currently, the following entities provide status information:

- GCC
- RT
- RM
- XM

4.1.4.1 Generic Call Control

The layer manager initiates the management - solicited status procedure to gather solicited status information about the GCC. The GCC status request primitive (**`CcMiLccStaReq`**) can be called more than once, any time after the management - configuration procedure.

The following GCC status request primitive types can be called.

- System ID
- Interface
- Observation trigger table
- Lower SAP status for the PSIF, RM, RT, and SF SAPs

The **`ccMgmnt.hdr.elmId`** field specifies the status request primitive type.

The status confirm (**`CcMiLccStaCfm`**) primitive is called during the status procedure, after the appropriate status structure has been initialized.

The **`ccMgmnt.t.ssta`** structure specifies parameters used by the status request and status confirm (**`CcMiLccStaReq`** and **`CcMiLccStaCfm`**) primitives.

The data flow is:

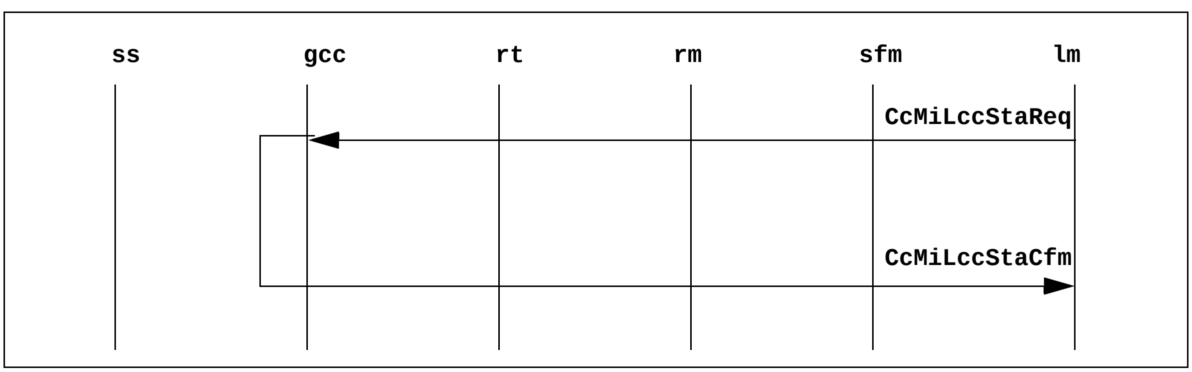


Figure 4-13Data flow: GCC solicited status procedure

4.1.4.2 Router

The layer manager initiates the management - solicited status procedure to gather solicited status information about the RT. The RT status request primitive (**RtMiLrtStaReq**) can be called more than once, any time after the management - configuration procedure.

The following RT status request primitive types can be called:

- Interface
- Route

The **rtMgmnt.hdr.elmId** field specifies the status request primitive type.

The status confirm (**RtMiLrtStaCfm**) primitive is called during the status procedure, after the appropriate status structure has been initialized.

The **rtMgmnt.t.ssta** structure specifies parameters used by the status request and status confirm (**RtMiLrtStaReq** and **RtMiLrtStaCfm**) primitives.

The data flow is:

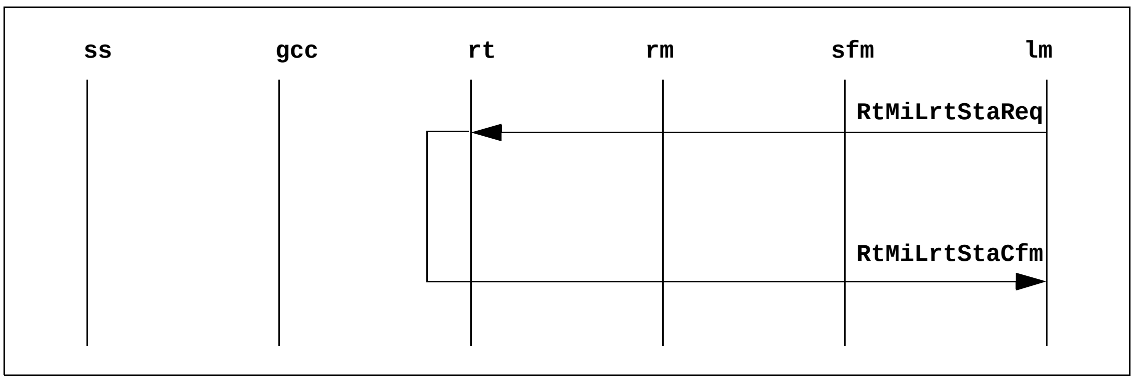


Figure 4-14Data flow: RT solicited status procedure

4.1.4.3 Resource Manager

The layer manager initiates the management - solicited status procedure to gather solicited status information about the RM. The RM status request primitive (**RtMiLrtStaReq**) can be called more than once, any time after the management - configuration procedure.

The following RM status request primitive types can be called.

- VPI (broadband only)
- VPI / VCI (broadband only)
- Circuit (ISUP only)
- DSS1 interface (ISDN only)
- DSS1 channel status (ISDN only)

The **rmMgmnt.hdr.elmId** field specifies the status request primitive type.

The status confirm (**RmMiLrmStaCfm**) primitive is called during the status procedure after the appropriate status structure has been initialized.

The **rmMgmnt.t.ssta** structure specifies parameters used by the status request and status confirm (**RmMiLrmStaReq** and **RmMiLrmStaCfm**) primitives.

The data flow is:

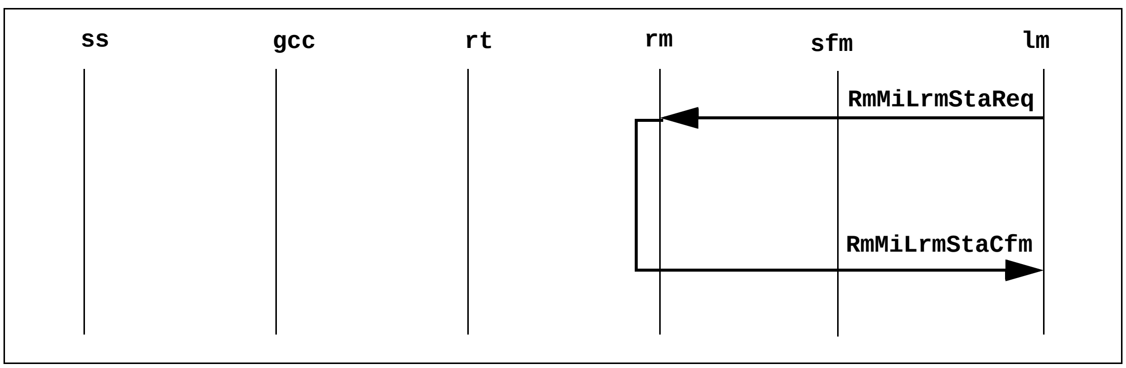


Figure 4-15Data flow: RM solicited status procedure

4.1.4.4 Connection Manager

The layer manager initiates the management - solicited status procedure to gather solicited status information about the XM. The XM status request primitive (**CcMiLccStaReq**) can be called more than once, any time after the management - configuration procedure.

The following XM status request primitive types can be called.

- Trunking IWF status
- AAL1 / AAL2 phase 2 trunking status
- VCCI status
- CID status
- CIC status
- ATM profile status
- VTOA profile status
- Signalling VCCI status
- VTOA phase 2 ATM resource status

The **xmMgmnt.hdr.elmId** field specifies the status request primitive type.

The status confirm (**XmMiLxmStaCfm**) primitive is called during the status procedure after the appropriate status structure has been initialized.

The **xmMgmnt.t.ssta** structure specifies parameters used by the status request and status confirm (**XmMiLxmStaReq** and **XmMiLxmStaCfm**) primitives.

The data flow is:

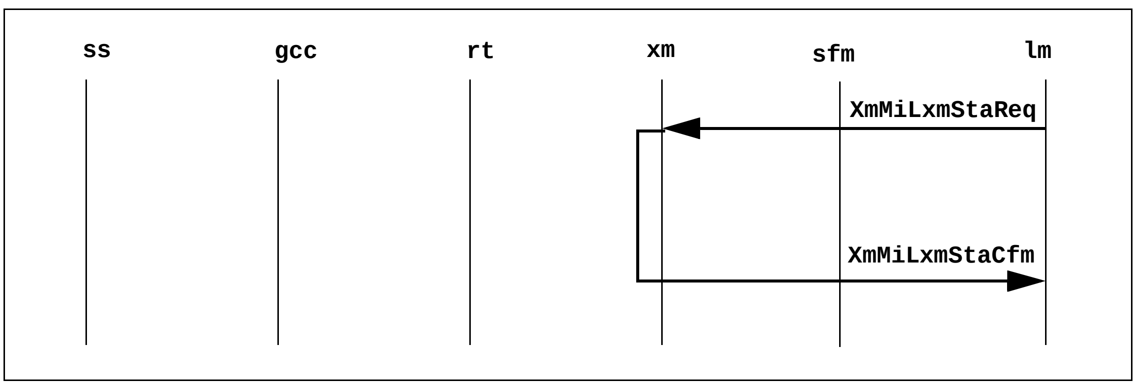


Figure 4-16Data flow: XM solicited status procedure

4.1.5 Management - Unsolicited Status

The management - unsolicited status procedure presents information about unsolicited status and alarms to the layer manager. The unsolicited status can be enabled or disabled via a control request.

4.1.5.1 Generic Call Control

GCC initiates this procedure. The status indication primitive (**CcMiLccStaInd**) can be called more than once any time after the configuration procedure, if the unsolicited status has been enabled. The status indication primitive is not called if the unsolicited status has been disabled. The unsolicited status can be enabled or disabled with the management - control procedure. For a description of the possible status indication values, see Section 3.5.1.2.9, "**CcMiLccStaInd**."

The system services primitives are called during the unsolicited status procedure if a loosely coupled layer manager interface is used.

The **CcMgmnt.t.usta** structure specifies parameters used by the status indication (**CcMiLccStaInd**) primitive.

The data flow is:

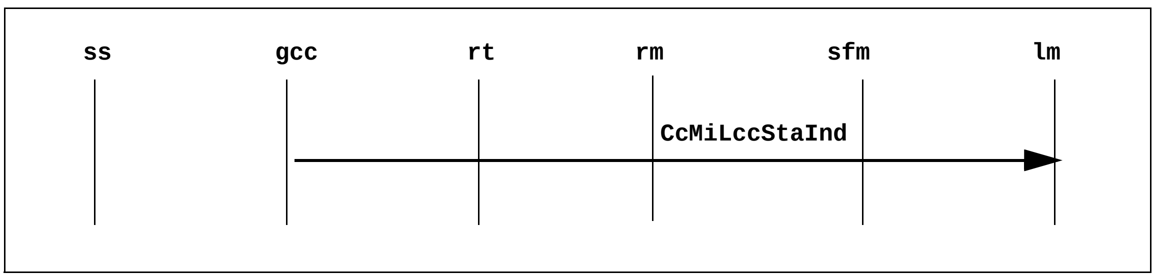


Figure 4-17Data flow: GCC unsolicited status procedure

4.1.5.2 Router

The RT initiates this procedure. The status indication primitive (**`RtMiLrtStaInd`**) may be called more than once any time after the configuration procedure, if the unsolicited status has been enabled. The status indication primitive is not called if the unsolicited status has been disabled. The unsolicited status can be enabled or disabled with the management - control procedure. For a description of the possible status indication values, see Section 3.6.1.2.9, "**`RtMiLrtStaInd`**."

The system services primitives are called during the unsolicited status procedure if a loosely coupled layer manager interface is used.

The **`RtMgmnt.t.usta`** structure specifies parameters used by the status indication (**`RtMiLrtStaInd`**) primitive.

The data flow is:

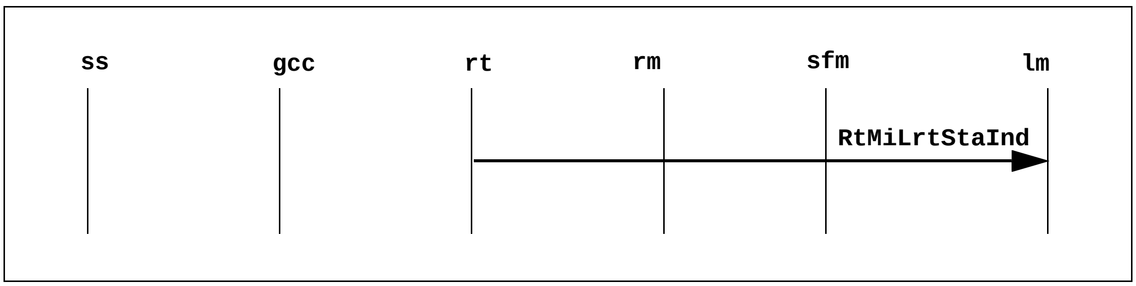


Figure 4-18Data flow: RT unsolicited status procedure

4.1.5.3 Resource Manager

The RM initiates this procedure. The status indication primitive (**RmMiLrmStaInd**) can be called more than once. The status indication primitive can be called any time after the configuration procedure, if the unsolicited status has been enabled. The status indication primitive is not called if the unsolicited status has been disabled. The unsolicited status can be enabled or disabled with the management - control procedure.

The system services primitives are called during the unsolicited status procedure if a loosely coupled layer manager interface is used.

The **RmMgmnt.t.usta** structure specifies parameters used by the status indication (**RmMiLrmStaInd**) primitive.

The data flow is:

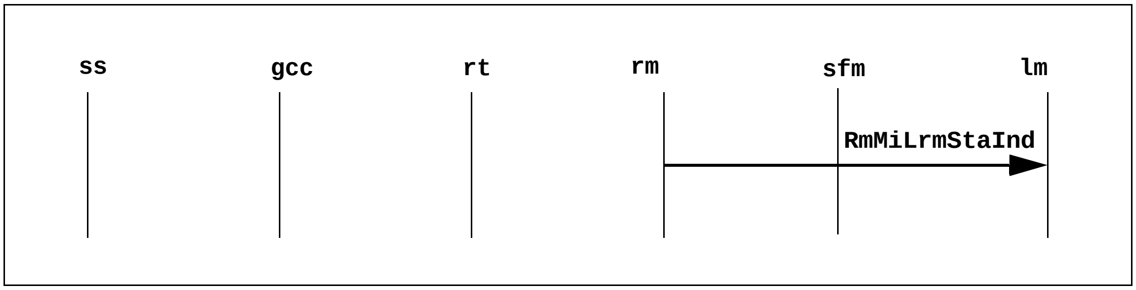


Figure 4-19Data flow: RM unsolicited status procedure

4.1.5.4 Connection Manager

The XM initiates this procedure. The status indication primitive (**XmMiLxmStaInd**) can be called more than once. The status indication primitive can be called any time after the configuration procedure, if the unsolicited status has been enabled. The status indication primitive is not called if the unsolicited status has been disabled. The unsolicited status can be enabled or disabled with the management - control procedure.

The system services primitives are called during the unsolicited status procedure if a loosely coupled layer manager interface is used.

The **XmMgmnt.t.usta** structure specifies parameters used by the status indication (**XmMiLxmStaInd**) primitive.

The data flow is:

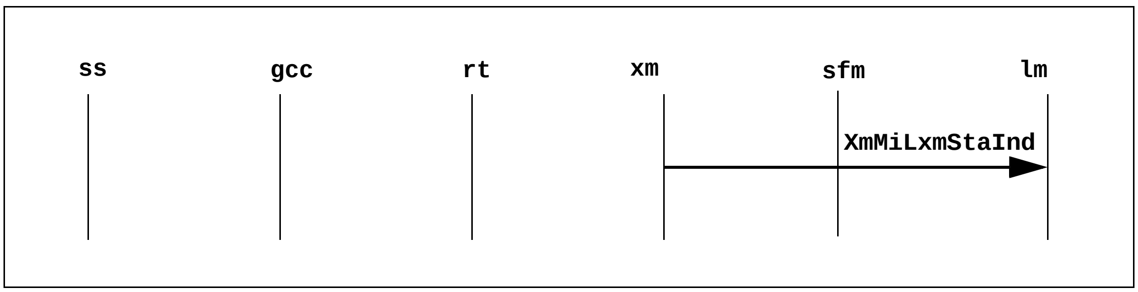


Figure 4-20Data flow: XM unsolicited status procedure

4.1.5.5 Switching Fabric Manager

The SFM initiates this procedure. The status indication primitive (**`SfMiLsfStaInd`**) can be called more than once any time after the configuration procedure, if the unsolicited status has been enabled. The status indication primitive is not called if the unsolicited status has been disabled. The unsolicited status can be enabled or disabled with the management - control procedure.

The system services primitives are called during the unsolicited status procedure if a loosely coupled layer manager interface is used.

The **`SfMgmnt.t.usta`** structure specifies parameters used by the status indication (**`SfMiLsfStaInd`**) primitive.

The data flow is:

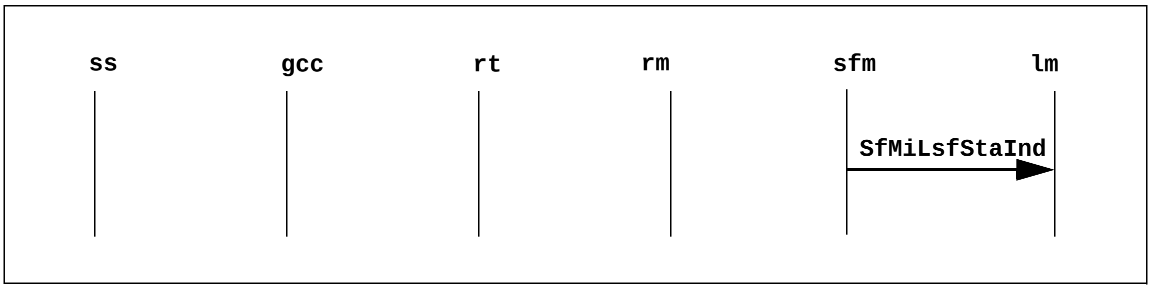


Figure 4-21Data flow: SFM unsolicited status procedure

4.1.6 Management - Control

The layer manager uses the management - control procedure to control the various elements of ICC. Currently, the following entities provide control requests:

- GCC
- RT
- RM
- XM

4.1.6.1 Generic Call Control

The layer manager initiates the management - control procedure to control GCC. The control request primitive (**CcMiLccCntrlReq**) can be called more than once. The GCC control request primitive can be called any time after the management - configuration procedure.

The following GCC control request primitive types can be called.

- Stack start, to start the interworking stack
- Enable or disable unsolicited status
- Enable or disable debug flags
- Clear an existing connection
- Initiate test call
- Enable, disable, or delete an interface
- Enable, disable, or delete virtual interface
- Enable or disable trace indications
- Enable or disable accounting indications
- Bind enable, unbind disable, gracefully disable, or delete a PSIF SAP
- Bind enable, unbind disable, gracefully disable, or delete a group of PSIF SAPs
- Bind enable, unbind disable, or delete an RM, RT, or SFM SAP
- Bind enable, unbind disable, or delete a group of RM, RT, or SFM SAPs
- Shut down the GCC layer

The **CcMgmnt.t.cntrl.type** field specifies the control request primitive type.

The system services primitives are called during the control procedure if a loosely coupled layer manager interface is used.

The **CcMgmnt.t.cntrl** structure specifies parameters used by the control request (**CcMiLccCntrlReq**) primitive.

The data flow is:

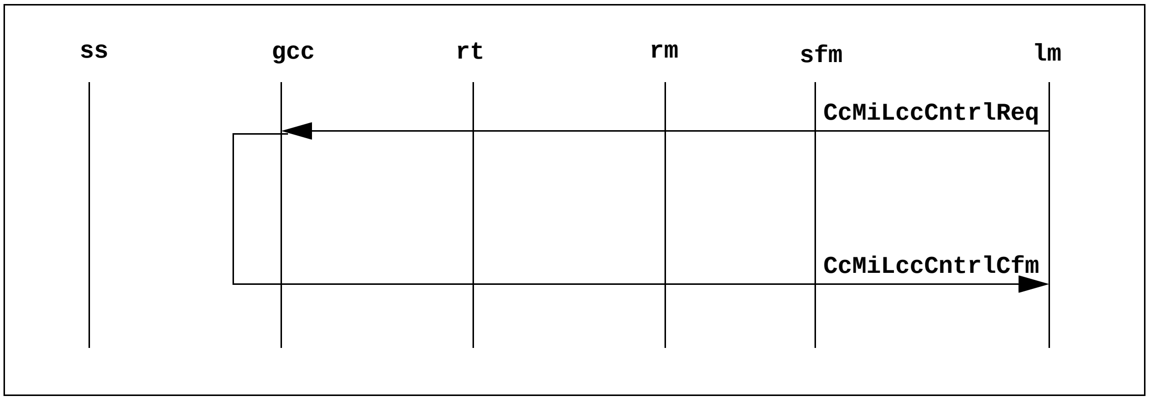


Figure 4-22Data flow: GCC control procedure

4.1.6.2 Router

The layer manager initiates the management - control procedure to control the RT. The control request primitive (**`RtMiLrtCntrlReq`**) can be called more than once, any time after the management - configuration procedure.

The following RT control request primitive types can be called.

- Enable or disable the unsolicited status
- Enable or disable the debug flags
- Delete route
- Enable, disable, or delete an interface
- Enable, disable, or delete a virtual interface
- Enable or disable unsolicited status
- Enable or disable debug flags
- Set the congestion level
- Set the observation index
- *Unbind disable* or delete an upper SAP
- *Unbind disable* a group of upper SAPs
- Enable or disable the periodic auditing procedures
- Enable or disable the one-time auditing procedures
- Shut down the RT layer

The **`RtMgmnt.t.cntrl.type`** field specifies the control request primitive type.

The system services primitives are called during the control procedure if a loosely coupled layer manager interface is used.

The **`RtMgmnt.t.cntrl`** structure specifies the parameters used by the control request (**`RtMiLrtCntrlReq`**) primitive.

The data flow is:

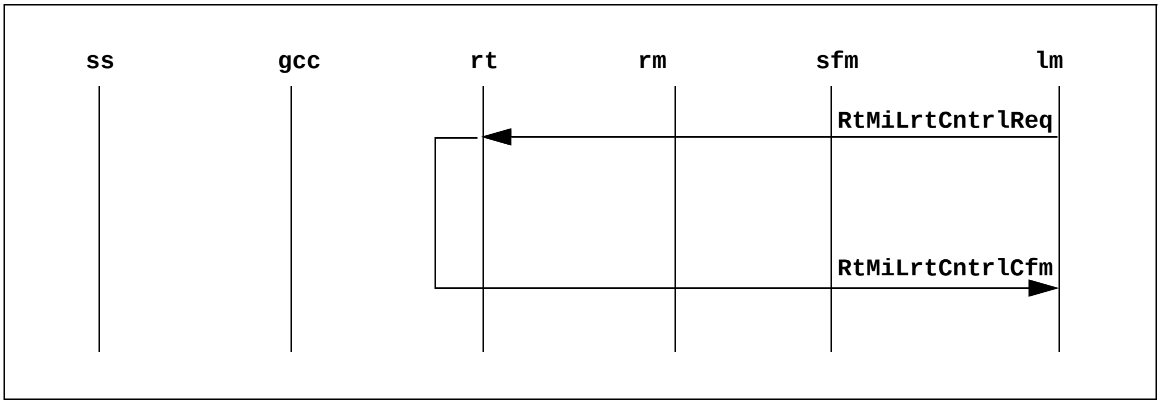


Figure 4-23Data flow: RT control procedure

4.1.6.3 Resource Manager

The layer manager initiates the management - control procedure to control the RM. The control request primitive (**`RmMiLrmCntrlReq`**) can be called more than once, any time after the management - configuration procedure.

The following RM control request primitive types can be called.

- Enable or disable the unsolicited status
- Delete a physical broadband link
- Delete a narrowband DPC or broadband interface
- Delete a VPI
- Enable or disable a VPI
- Reset a VPI / VCI
- Delete a circuit
- Enable, disable, or reset a circuit
- Delete a static binding
- Enable or disable unsolicited status
- Enable or disable debug flags
- Delete a DSS1 configured interface
- Disable the specified channels of a DSS1 interface
- Enable the specified channels of a DSS1 interface
- Reset the specified channels of a DSS1 interface
- Equip the specified channels with a DSS1 interface
- Do not equip the specified channels of a DSS1 interface
- Set an observation index
- *Unbind disable* or delete an upper SAP
- *Unbind disable* a group of upper SAPs
- Enable or disable the periodic auditing procedures
- Enable or disable the one-time auditing procedures
- Enable or disable the GCC auditing procedures
- Shut down the RM layer

The **`RmMgmnt.t.cntrl.type`** field specifies the control request primitive type.

The system services primitives are called during the control procedure if a loosely coupled layer manager interface is used.

The **`RmMgmnt.t.cntrl`** structure specifies the parameters used by the control request (**`RmMiLrmCntrlReq`**) primitive.

The data flow is:

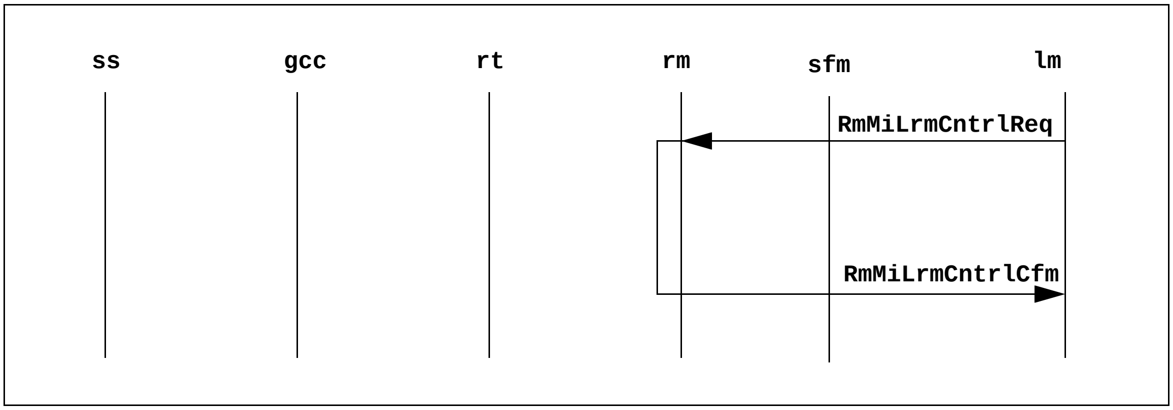


Figure 4-24Data flow: RM control procedure

4.1.6.4 Connection Manager

The layer manager initiates the management - control procedure to control the XM. The control request primitive (**`XmMiLxmCntrlReq`**) can be called more than once, any time after the management - configuration procedure.

The following XM control request primitive types can be called.

- Enable or disable alarms
- Enable or disable debug classes
- Delete an SAP
- Delete an IWF control block
- Delete a VCCI
- Make a VCCI available or unavailable for connections
- Delete a signalling VCCI
- Make a signalling VCCI available or unavailable for connections
- Delete a CIC
- Make a CIC available or unavailable for connections
- Delete a CID
- Make a CID available or unavailable for connections
- Delete an ATM profile
- Delete a VTOA profile

The **`XmMgmnt.t.cntrl.type`** field specifies the control request primitive type.

The system services primitives are called during the control procedure if a loosely coupled layer manager interface is used.

The **`XmMgmnt.t.cntrl`** structure specifies the parameters used by the control request (**`XmMiLxmCntrlReq`**) primitive.

The data flow is:

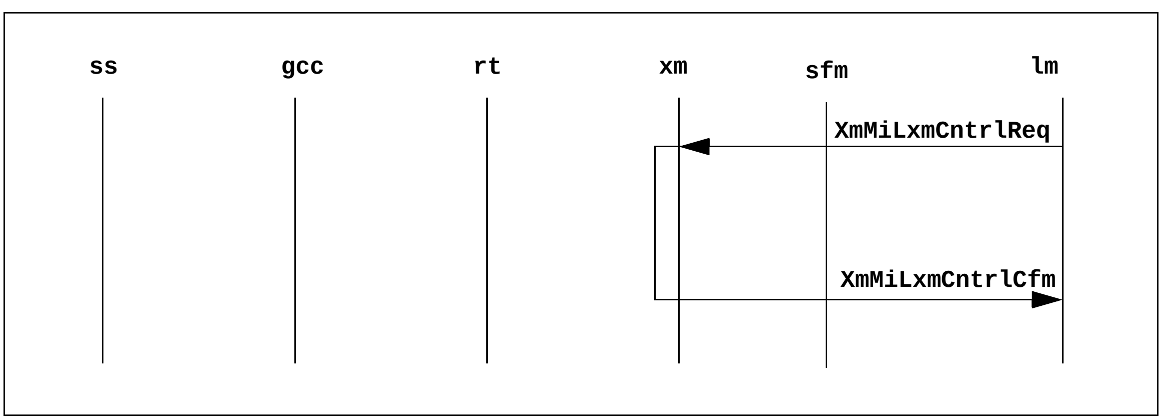


Figure 4-25Data flow: XM control procedure

4.1.6.5 Management - Control: Bind

A specific ICC layer manager control request, the stack start request toward GCC (**CcMiLccCntrlReq**), is a one-shot command for activating the entire signalling stack.

The command **LCC_STKSTART** forces GCC to bind all the lower layers.

The system services primitives are called during the control procedure if a loosely coupled layer manager interface is used.

The **CcMgmnt.t.cntrl** structure specifies parameters used by the control request (**CcMiLccCntrlReq**) primitive.

The data flow is:

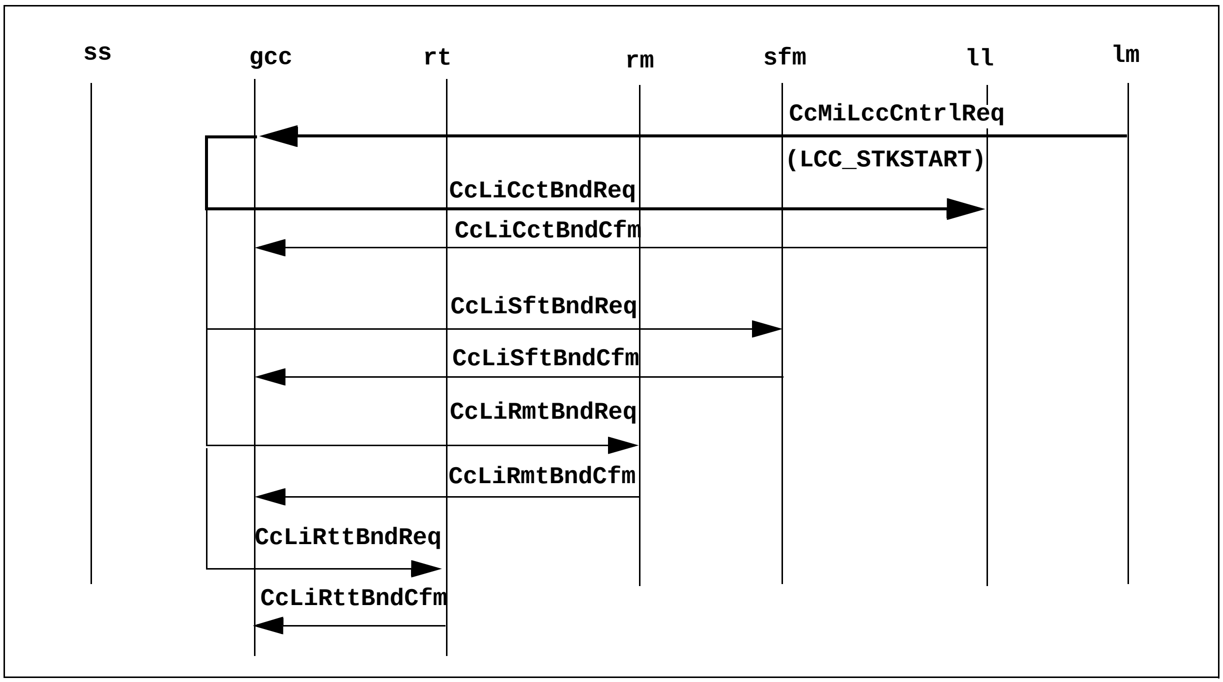


Figure 4-26Data flow: Bind procedure

4.2 VCCIs for Trunking

This section describes VCCI trunking assigned to each virtual connection between two IWFs. The channel ID in the Q.931 messages carries the VCCI and CID for an ISDN call. There is a one-to-one correspondence between a trunking VCCI and an ISDN Non-Facility Associated Signalling (NFAS) link, at an ISDN signalling interface in Q.931—Q.931 must know about the VCCIs. In case of PVC, in which all the VCCIs are preconfigured in ICC, corresponding interfaces must be configured in Q.931.

For the SVC (dynamic setup of VCCs), the interface must be configured and enabled in Q.931 when a VCC is set up and a VCCI is allocated. Since there is no primitive at the Q.931 upper interface for configuration, it must be done via the stack manager. The XM sends an indication to the stack manager when a VCC is set up and a VCCI is allocated. Then, the stack manager must configure the corresponding interface in Q.931 and re-send an indication to the XM when Q.931 can handle traffic at the new interface.

Note: *The confirmation is not sent to the XM before the interface is set up and can transport data.*

There are two different cases:

- Signalling VCCIs (phase 1 trunking)
- Bearer VCCIs (phase 1 trunking and feature transparency)

4.2.1 Establishing the Signalling VCCI

Signalling VCCs are virtual links (tunnels) through a network that tunnels Q.931 signalling. In case of phase 1 trunking, this signalling is carried over ATM links. When a signalling VCC is set up and a signalling VCCI is allocated, the corresponding NFAS link must be configured in Q.931 and the stack manager must make the association between the NFAS link and VPI/VCI.

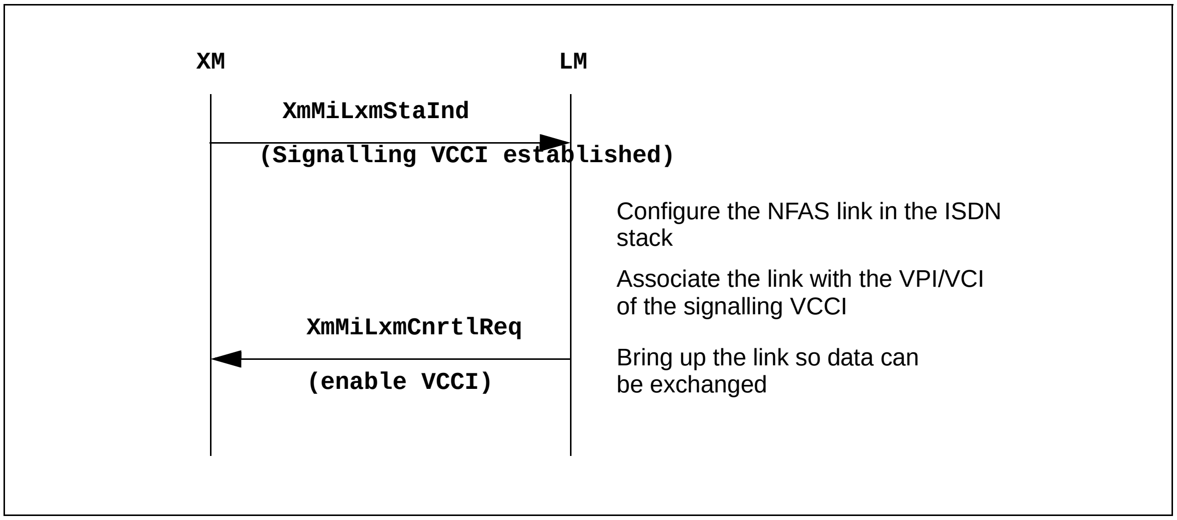


Figure 4-27Setting up a signalling VCCI

The event and cause passed in the status indication depend on the type of AAL used.

AAL Type	Event	Cause
AAL 1	**`LXM_EVENT_AAL1_SIGSVC`**	**`LXM_CAUSE_SIGVCCESTABLISHED`**
AAL 2	**`LXM_EVENT_AAL2_SIGSVC`**	**`LXM_CAUSE_SIGVCCESTABLISHED`**

The characteristics of the ATM connection is passed in the **`aalConParam`** of the SVC information field.

The following steps, in order, are for Trillium’s protocol stack procedure.

1. Status indication from the XM to the stack manager indicates that a signalling VCC has been established. The VCCI is passed to the stack manager.
2. Configures the link in Q.930 / Q.931.
3. Configures the interface in PSIF - Q.930 / Q.931.
4. Configures the upper and lower SAPs in Q.930 / Q.931, over the Q.SAAL convergence layer.
5. The stack manager programs the association between the Q.930 / Q.931 link and VPI / VCI, in the Q.930 / Q.931-to-Q.SAAL convergence layer. The convergence layer then triggers Q.SAAL to bring up the link.
6. The stack manager gives a confirmation to the XM again, which starts the Q.931 signalling procedures over the newly established signalling VCC.

Note: *The XM assumes that the link is ready for data transmission when the control request is sent.*

Figure 4-28 illustrates setting up signalling VCCs.

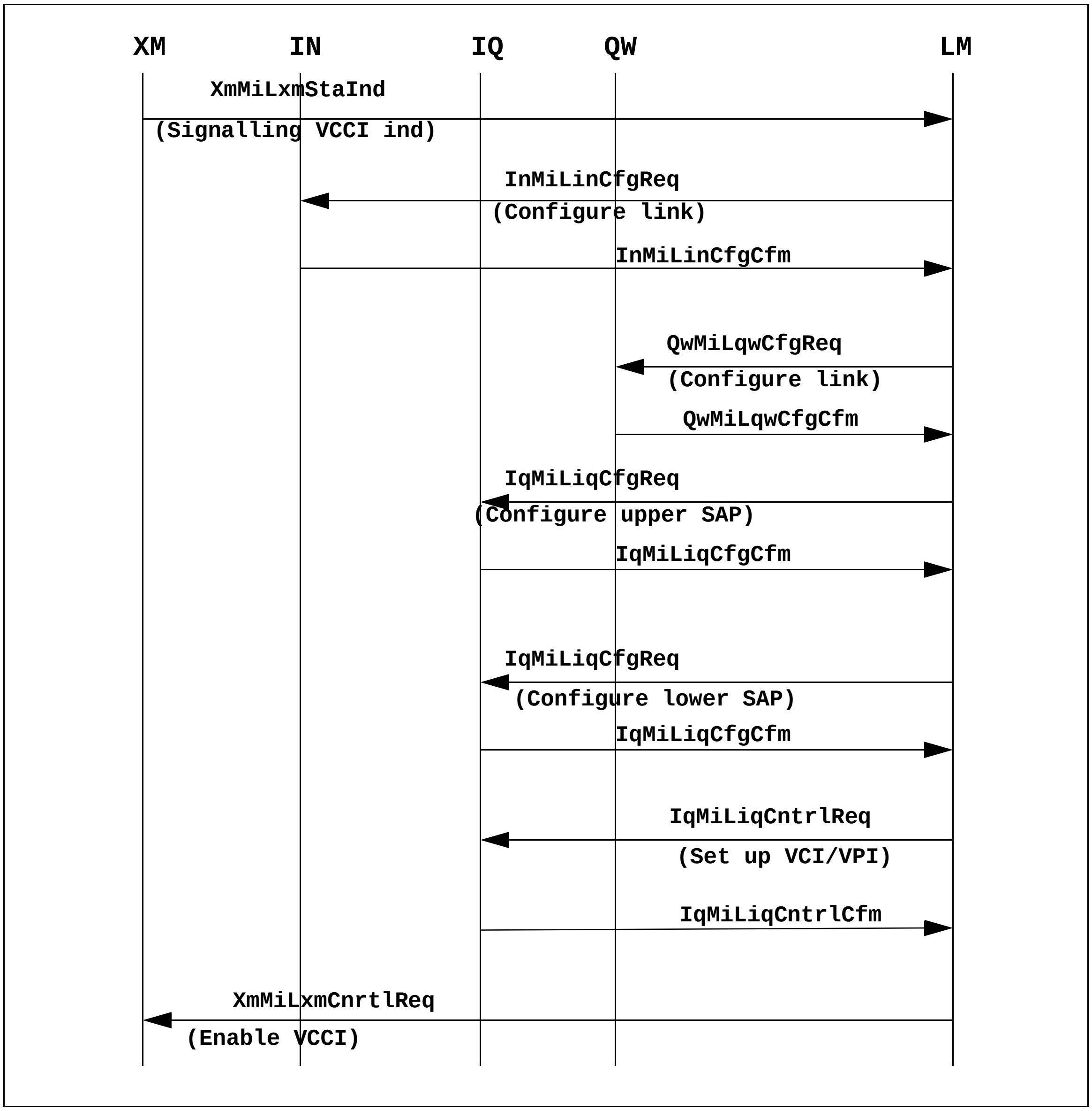


Figure 4-28Dynamic setup of signalling VCCs

4.2.2 Releasing the Signalling VCCI

Releasing signalling VCCs is an unacknowledged indication to the stack manager. The XM does not send any further data on the VCC, and therefore, does not wait for a confirmation of the VCC being released.

The data flow is:

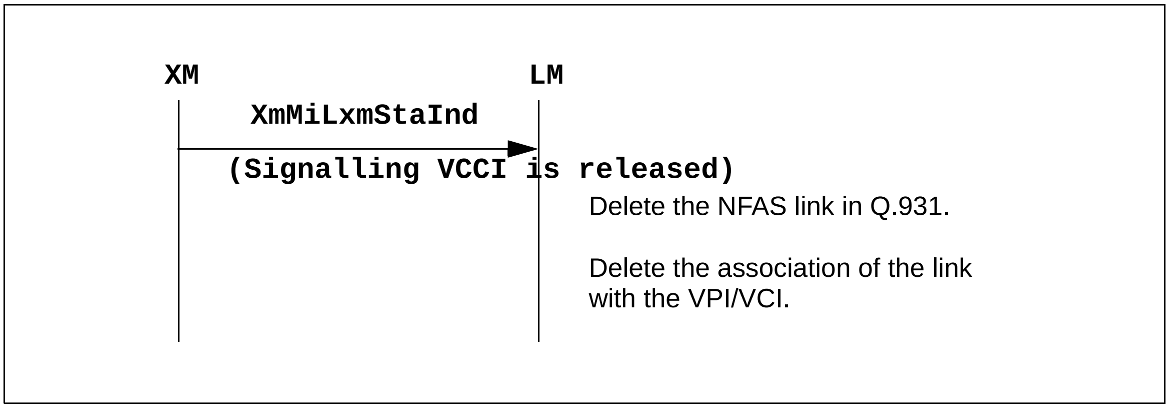


Figure 4-29Releasing the signalling VCCI

The following steps, in order, are for Trillium's protocol stack procedure.

1. Status indication from the XM to the stack manager indicates that a signalling VCC has been released. The VCCI is passed to the stack manager.
2. Deletes the link in Q.930 / Q.931.
3. Deletes the interface in PSIF - Q.930 / Q.931.
4. Deletes the upper and lower SAPs in Q.930 / Q.931, over the Q.SAAL convergence layer.

Figure 4-30 shows the release of signalling VCCs.

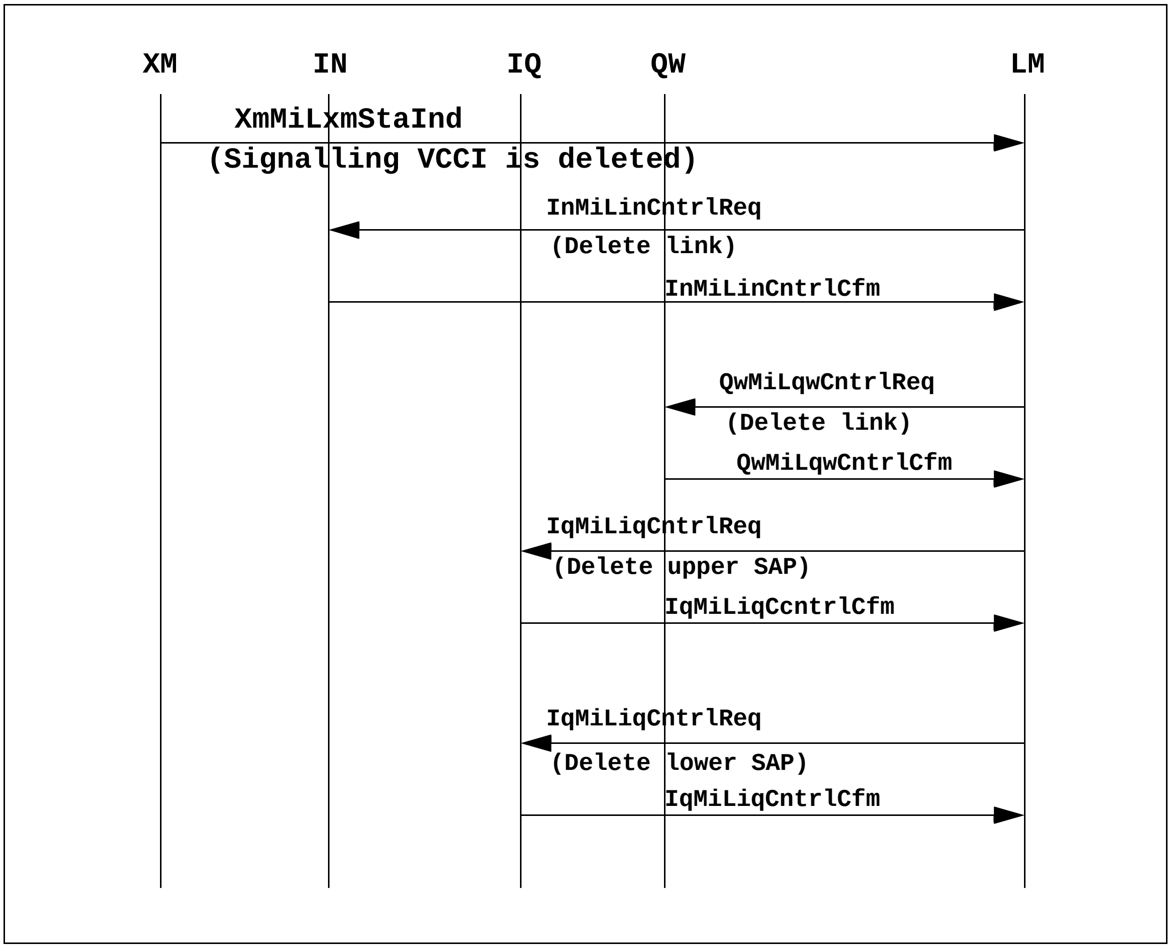


Figure 4-30Release of signalling VCCs

The event and cause passed in the status indication depend on the AAL type used.

AAL Type	Event	Cause
AAL 1	**LXM_EVENT_AAL1_SIGSVC**	**LXM_CAUSE_SIGVCCRELEASED**
AAL 2	**LXM_EVENT_AAL2_SIGSVC**	**LXM_CAUSE_SIGVCCRELEASED**

4.2.3 Establishing the Bearer VCCI

Bearer VCCs are virtual links that carry bearer traffic. For phase 1 trunking and feature transparency, Q.931 must know about each bearer VCC since each corresponds to an interface in Q.930 / Q.931. For each bearer VCC, a unique ID (VCCI) that has a one-to-one correspondence is allocated to an interface in Q.931; thus, for each allocated VCCI, the corresponding interface must be configured in Q.930 / Q.931.

The data flow is:

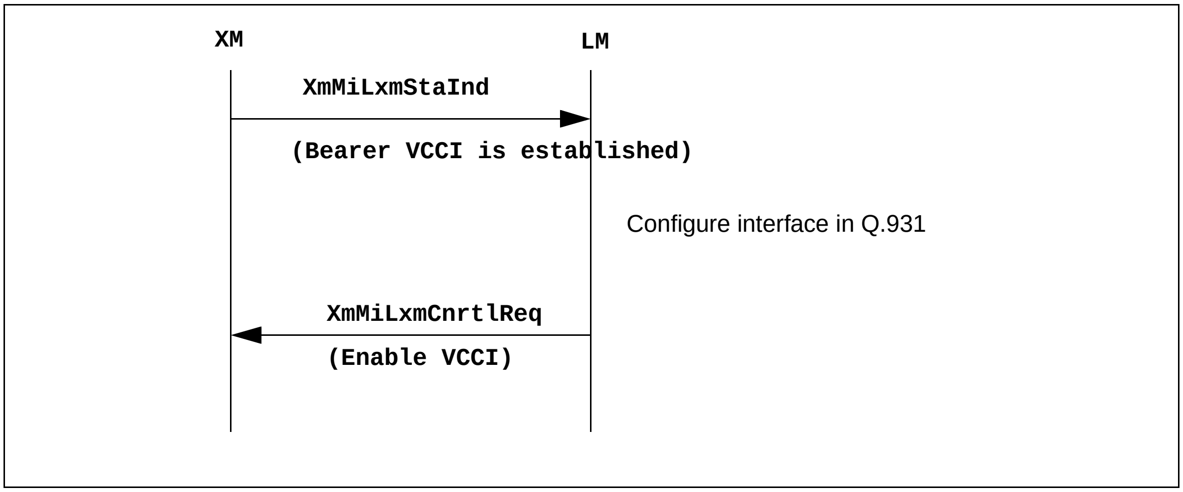


Figure 4-31Establishing the bearer VCC

The event and cause passed in the status indication depend on the AAL type used.

Type	Event	Cause
AAL 1	**LXM_EVENT_AAL1_BEARESVC**	**LXM_CAUSE_BEARERVCCESTABLISHED**
AAL 2	**LXM_EVENT_AAL2_BEARERSVC**	**LXM_CAUSE_BEARERVCCESTABLISHED**
Feature Transparency	**LXM_EVENT_FEATTRP_BEARERSVC**	**LXM_CAUSE_BEARERVCCESTABLISHED**

The following steps, in order, are for Trillium's protocol stack procedure.

- Status indication from the XM to the stack manager indicates that a signalling VCC has been established. The VCCI is passed to the stack manager.
- The stack manager configures the interface in Q.930 / Q.931.
- The stack manager gives a confirmation to the XM again. This starts the Q.931 signalling procedures over the newly established signalling VCC.

Note: *The XM assumes that the link is ready for data transmission when the control request is sent.*

The data flow is:

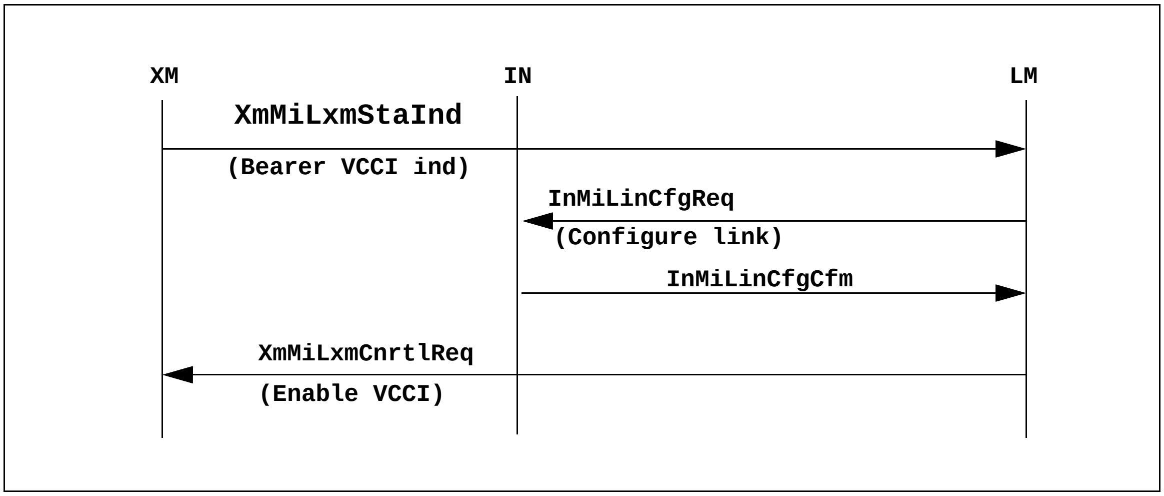


Figure 4-32Dynamic setup of bearer VCCs

4.2.4 Releasing the Bearer VCCI

Releasing bearer VCCs is an unacknowledged indication to the stack manager. The XM does not send further data on the VCC, and therefore, does not wait for a confirmation of the VCC being released.

The data flow is:

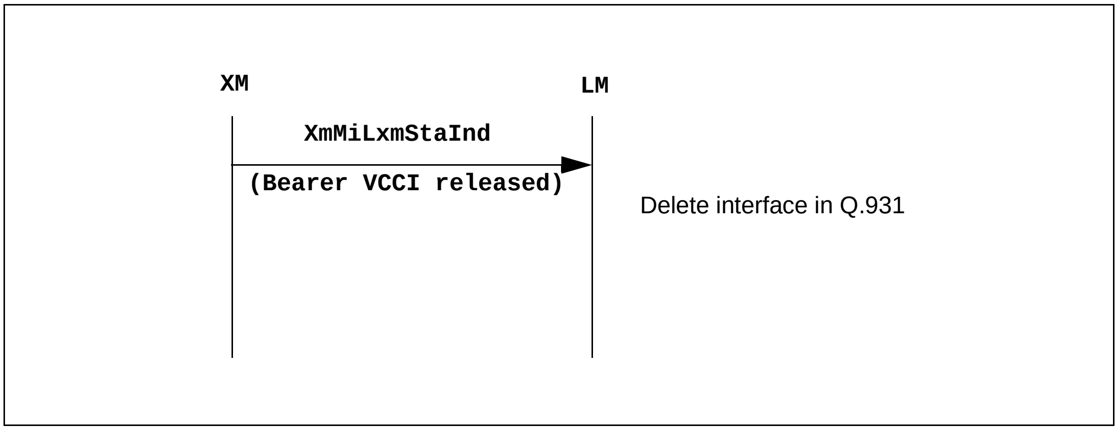


Figure 4-33Releasing the bearer VCCI

The following steps, in order, are for Trillium's protocol stack procedure.

- Status indication from the XM to the stack manager indicates that a signalling VCC has been released. The VCCI is passed to the stack manager.
- Deletes the link in Q.930 / Q.931.
- Deletes the interface in PSIF - Q.930 / Q.931.
- Deletes the upper and lower SAPs in Q.930 / Q.931, over the Q.SAAL convergence layer.

The data flow is:

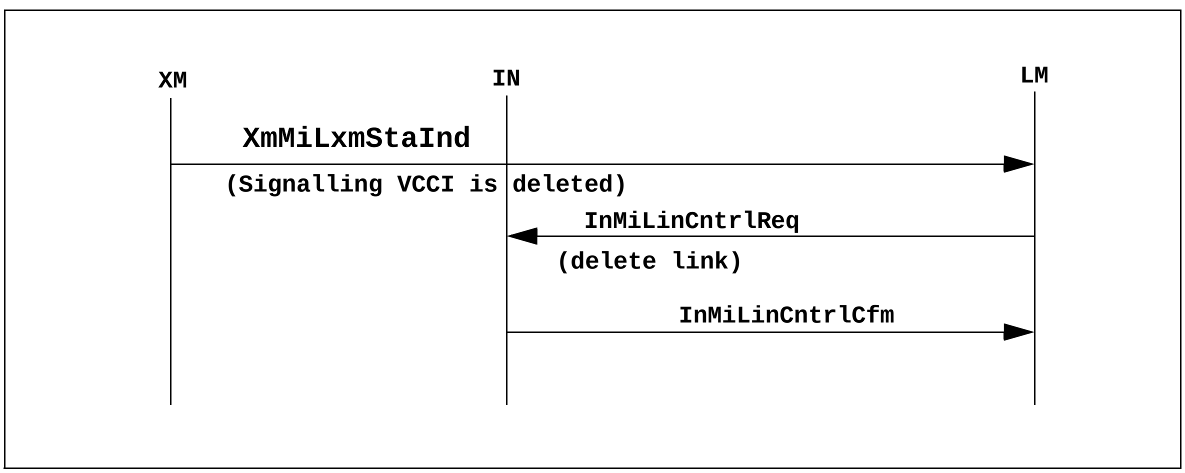


Figure 4-34Release of bearer VCCs

The event and cause passed in the status indication depend on the AAL type used.

AAL Type	Event	Cause
AAL 1	**LXM_EVENT_AAL1_BEARESVC**	**LXM_CAUSE_BEARERVCCRELEASED**
AAL 2	**LXM_EVENT_AAL2_BEARERSVC**	**LXM_CAUSE_BEARERVCCRELEASED**
Feature Transparency	**LXM_EVENT_FEATTRP_BEARERSVC**	**LXM_CAUSE_BEARERVCCRELEASED**

4.3 Basic Call Setup

The following call flow diagrams show the basic call setup between two different protocols.

4.3.1 Basic Call Flow: ISDN to ISUP

The data flow is:

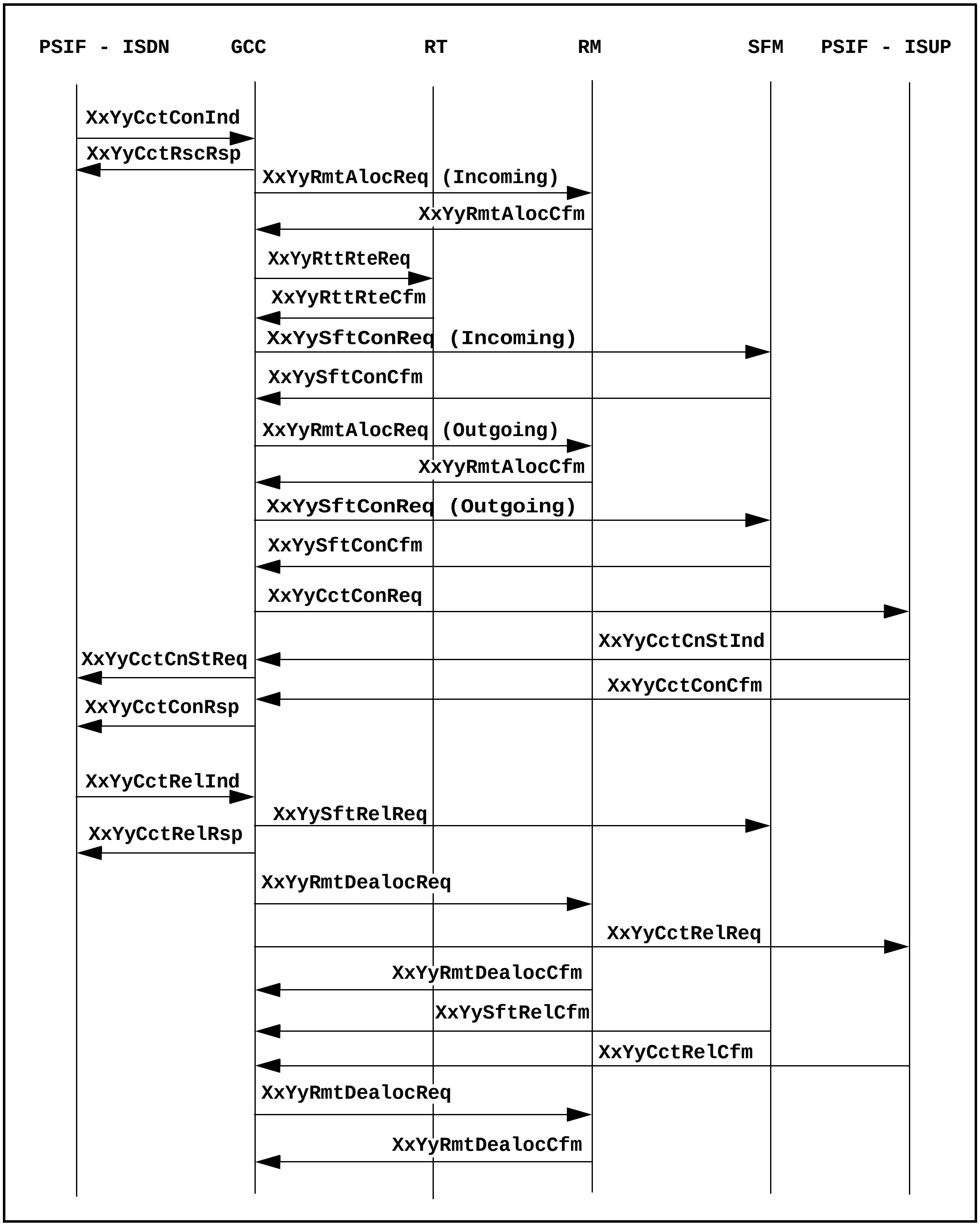


Figure 4-35Basic call flow: ISDN to ISUP

4.3.2 Basic Call Flow: ISUP to ISDN

The data flow is:

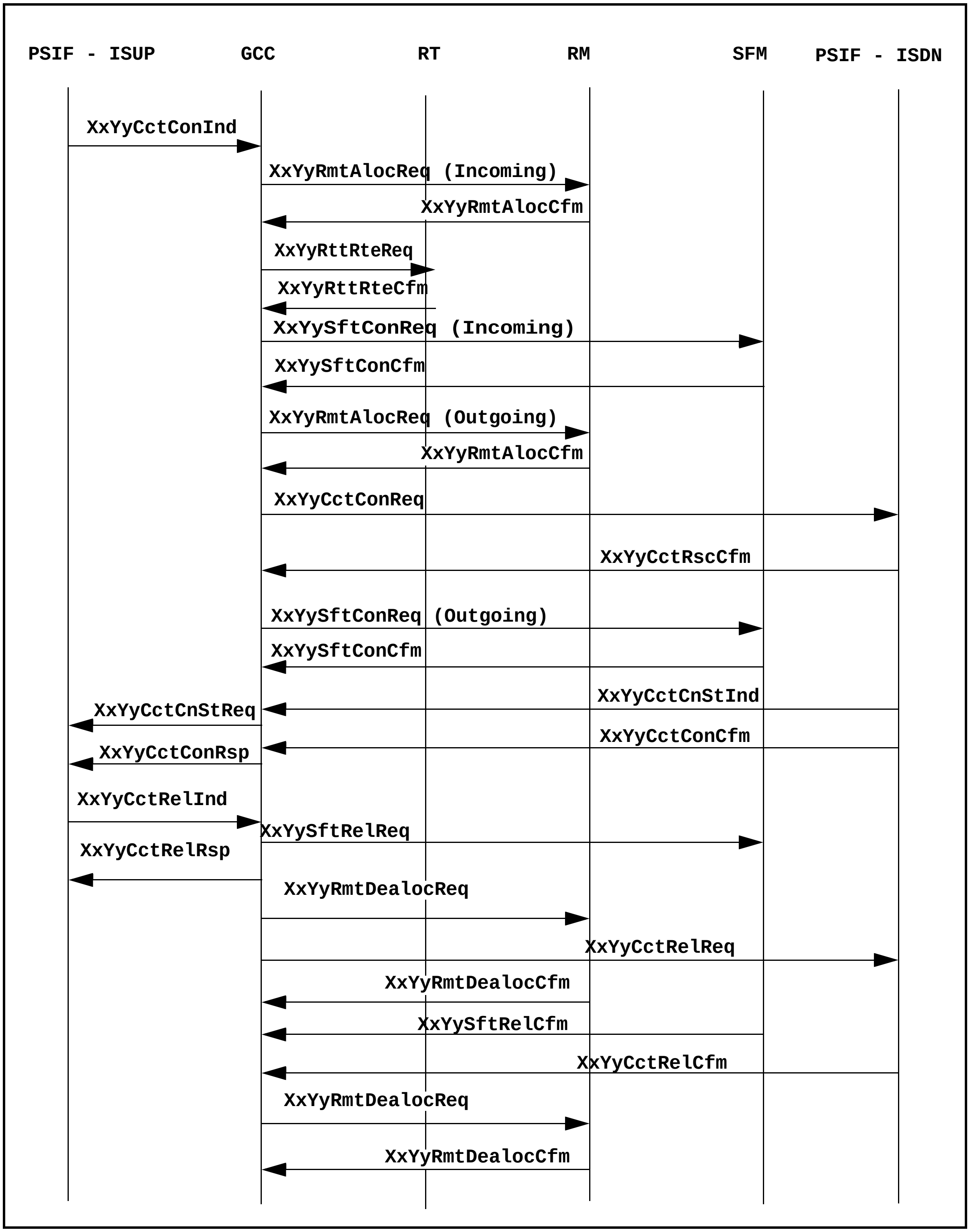


Figure 4-36Basic call flow: ISUP to ISDN

4.3.3 VTOA Phase 1 Trunking Using AAL2: Call Establishment at the Originating IWF

The data flow is:

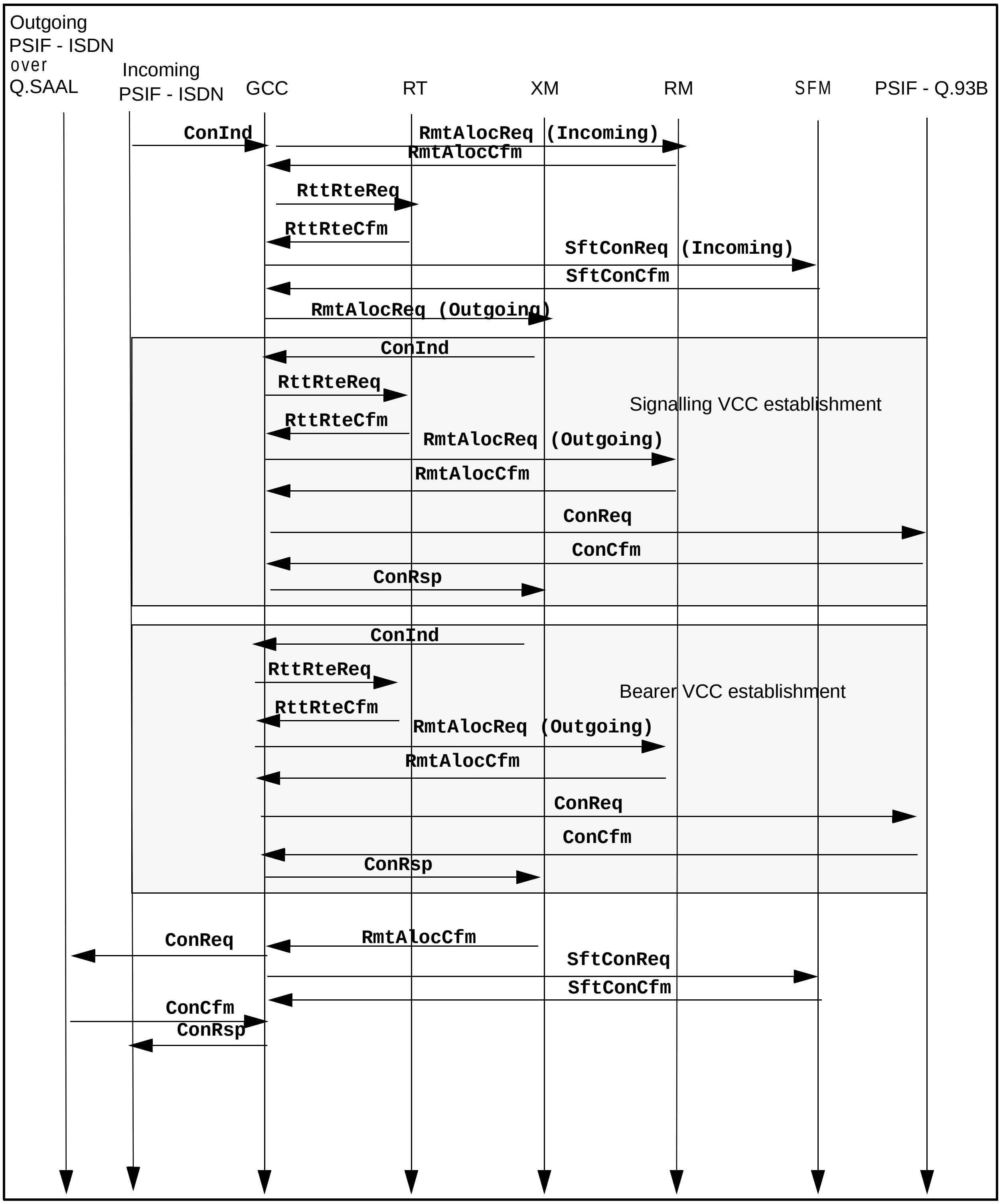


Figure 4-37: VTOA phase 1 trunking using AAL2: Call establishment at the originating IWF

4.3.4 VTOA Phase 1 Trunking Using AAL2: Call Establishment at the Terminating IWF

The data flow is:

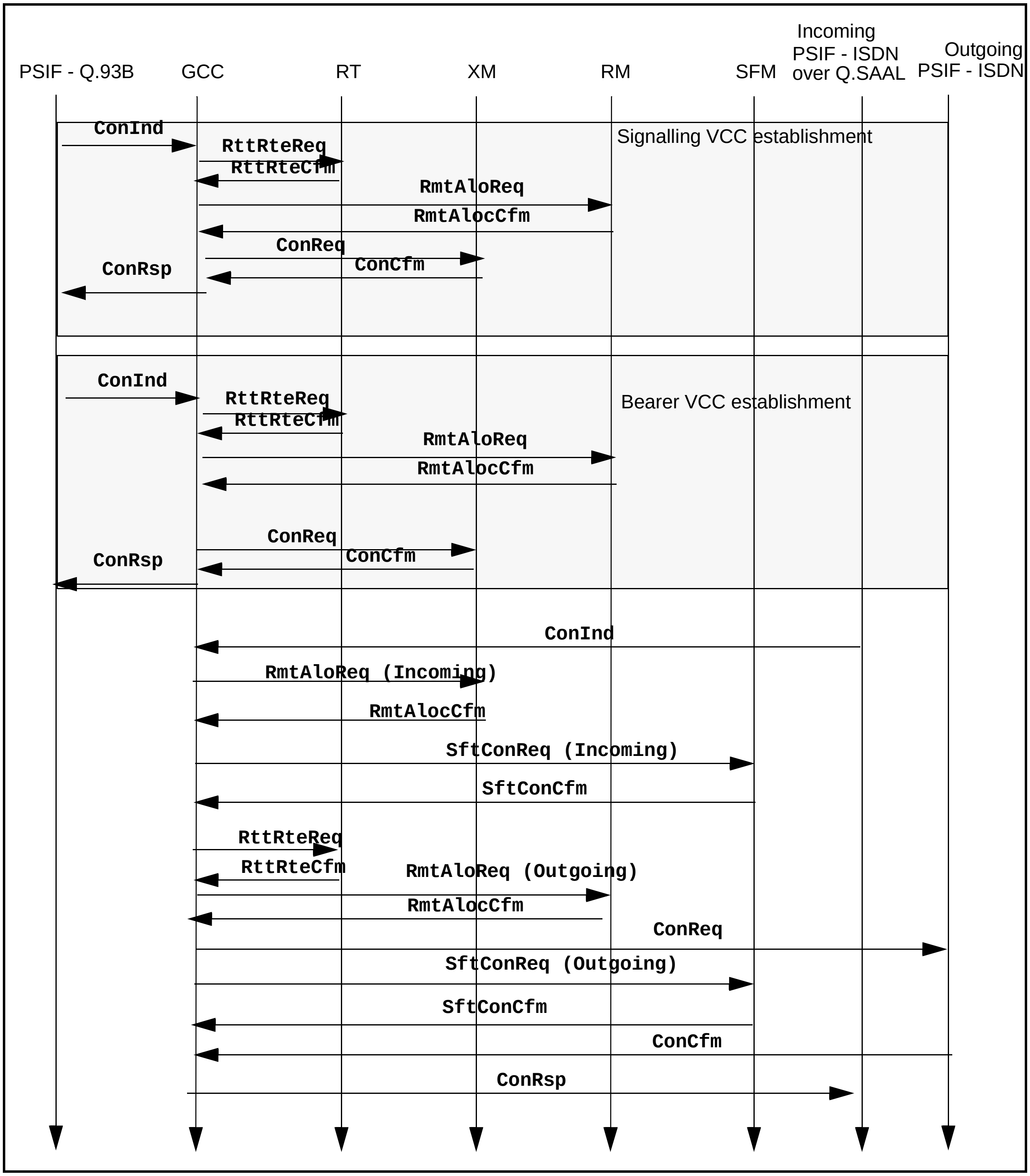


Figure 4-38: VTOA phase 1 trunking using AAL2: Call establishment at the terminating IWF

4.3.5 VTOA Phase 1 Trunking Using AAL2: Call Release at the Originating IWF

The data flow is:

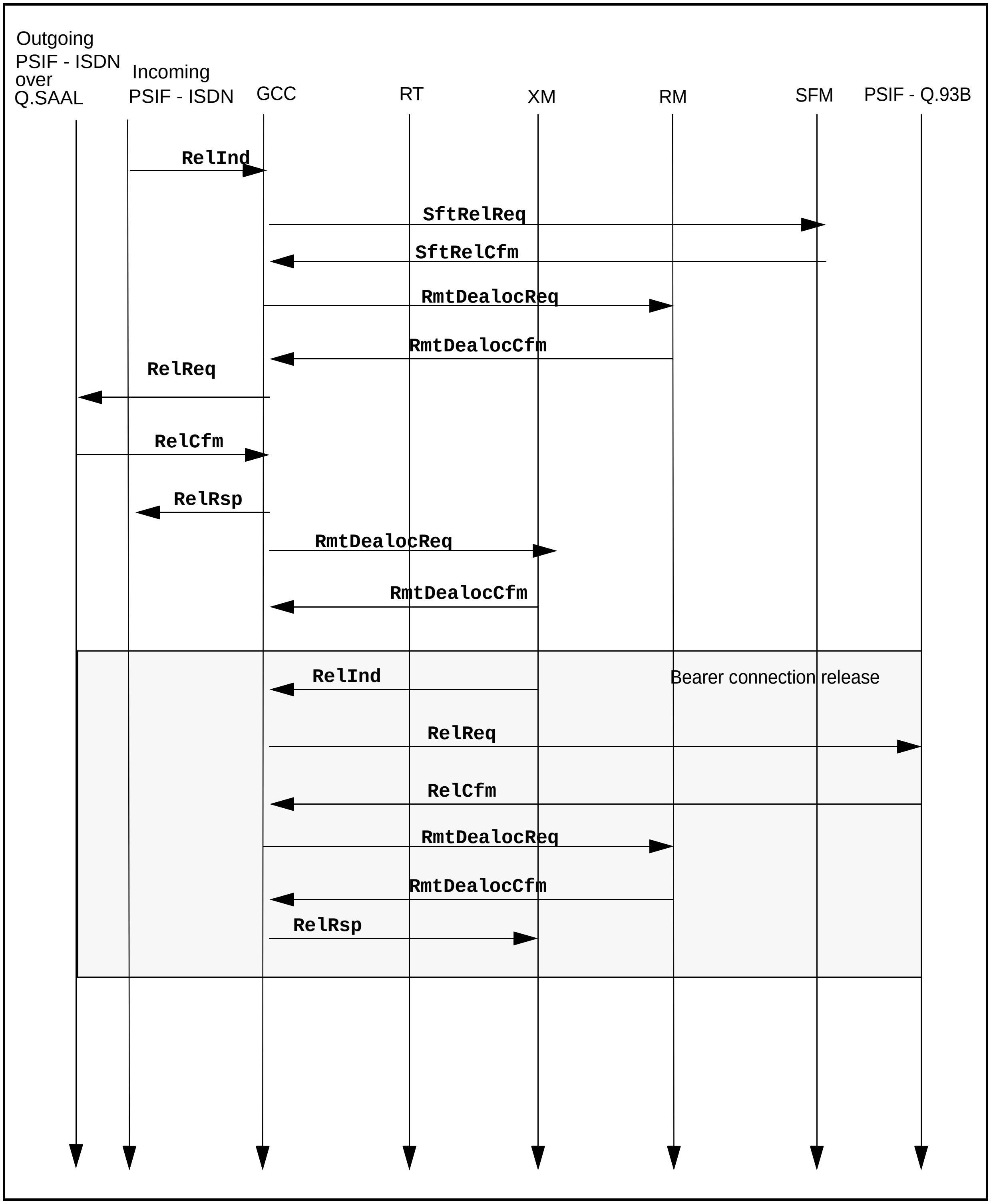


Figure 4-39: VTOA phase 1 trunking using AAL2: Call release at the originating IWF

4.3.6 VTOA Phase 1 Trunking Using AAL2: Call Release at the Terminating IWF

For a date flow illustration, see Section 4.3.5, "VTOA Phase 1 Trunking Using AAL2: Call Release at the Originating IWF."

4.3.7 Feature Transparency: Call Establishment at the Originating IWF

The data flow is:

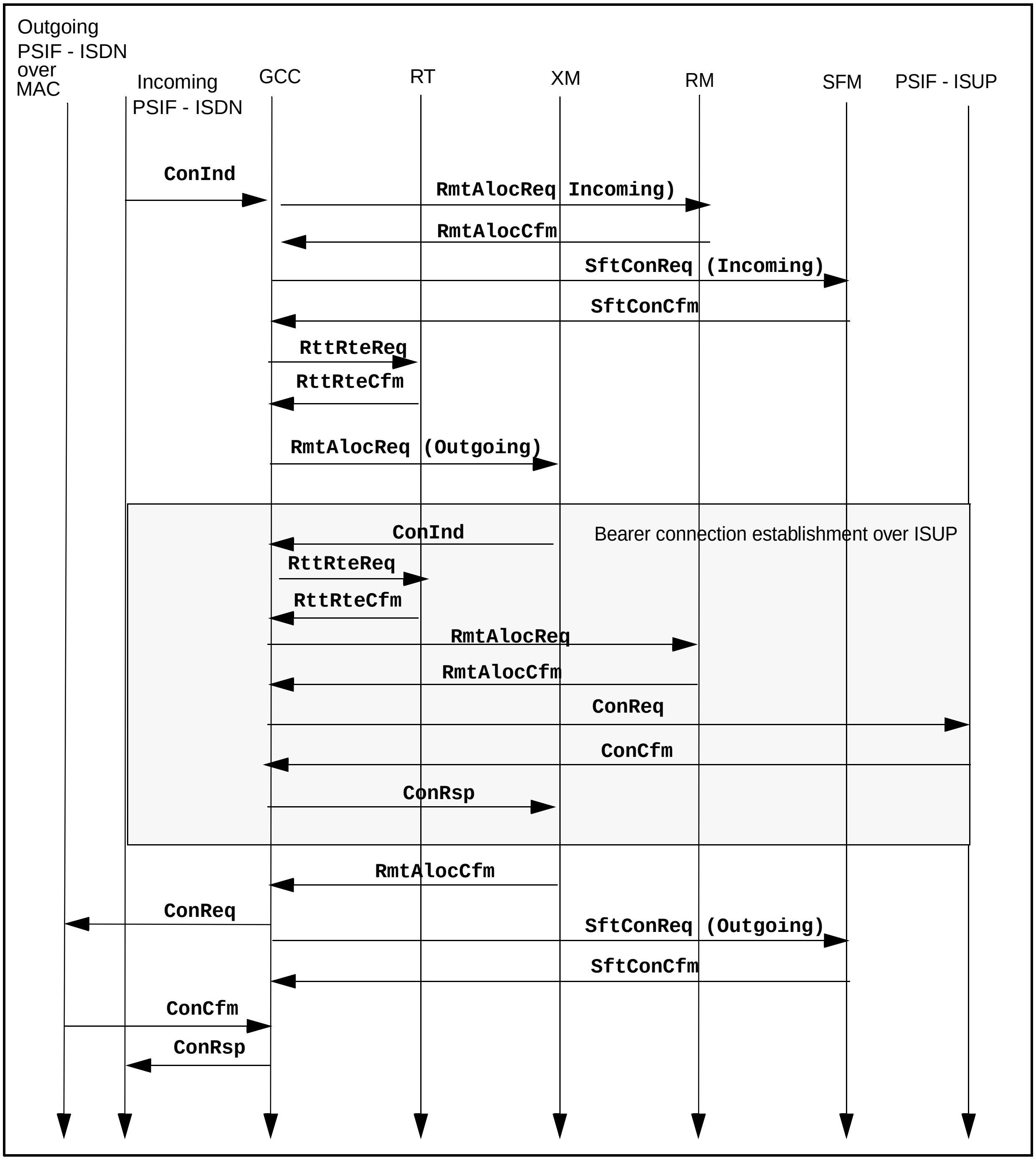


Figure 4-40: Feature transparency: Call establishment at the originating IWF

Note: *The shaded rectangles in Figures 4-45, 4-46, and 4-47 represent the areas in which the feature transparency call is either set up or tore down.*

4.3.8 Feature Transparency: Call Establishment at the Terminating IWF

The data flow is:

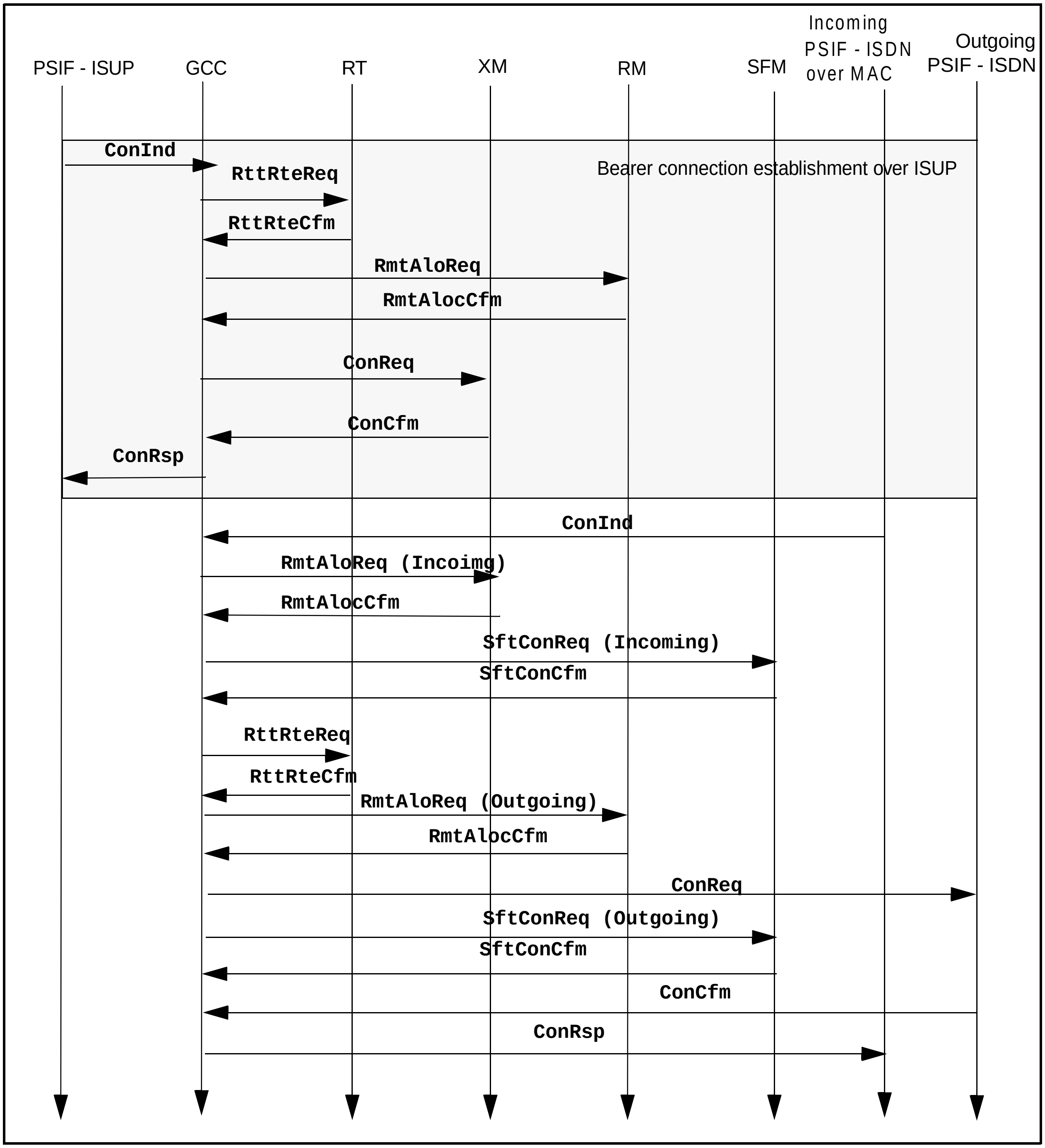


Figure 4-41: Feature transparency: Call establishment at the terminating IWF

4.3.9 Feature Transparency: Call Release at the Originating IWF

The data flow is:

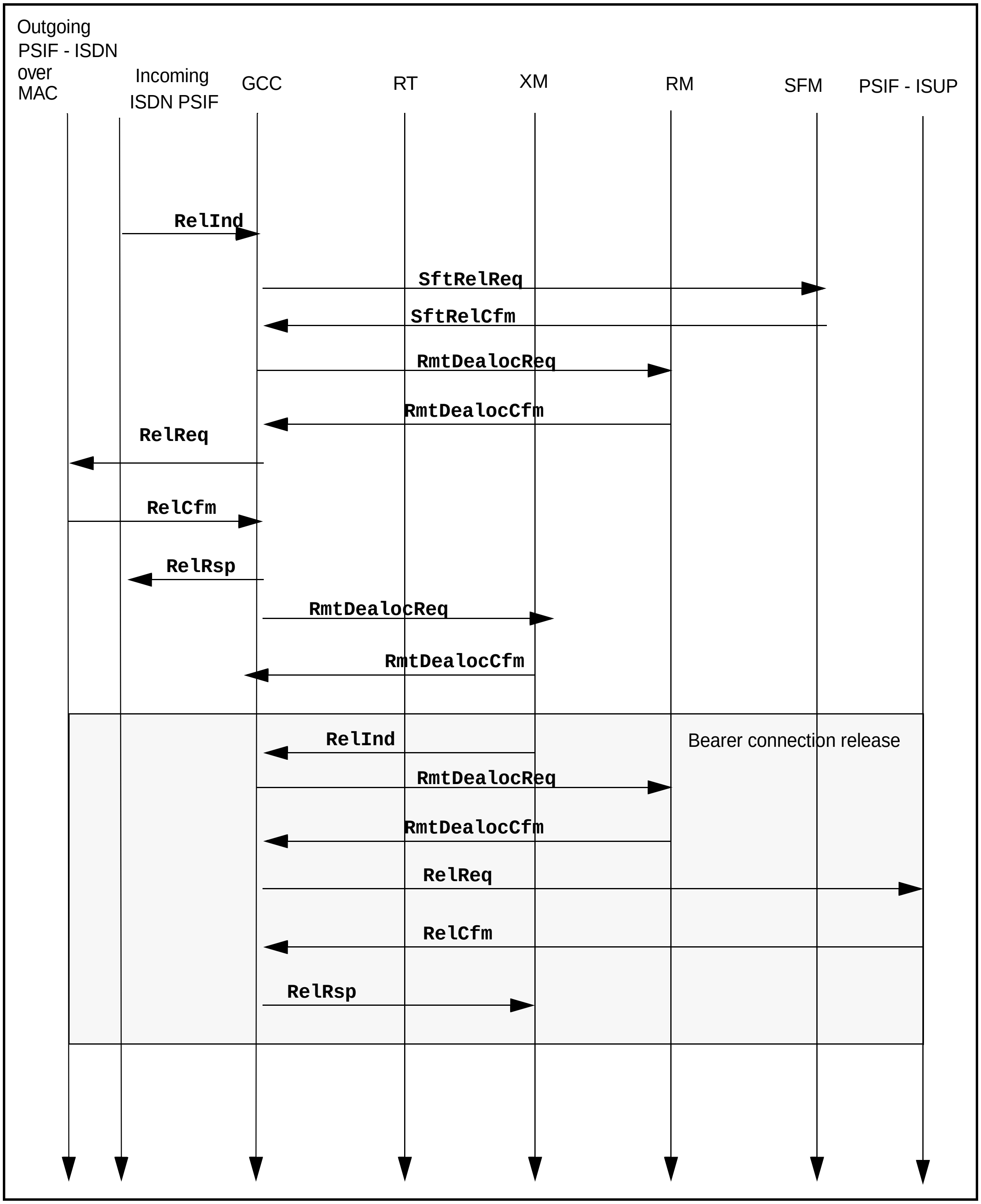


Figure 4-42: Feature transparency: Call release at the originating IWF

4.3.10 Feature Transparency: Call Release at the Terminating IWF

For a data flow illustration, see Section 4.3.9, "Feature Transparency: Call Release at the Originating IWF."

APPENDIX A: Broadband Profile

The format of the Broadband Profile structure is:

```
typedef struct ccBBProfCfg      /* Broadband Profile Configuration
                                   Structure */
{
   U8 profId;                   /* profile identifier */
   U8 profType;                 /* profile type */
   union
   {
      CcAtmParms  ituProf;      /* ITU BB profile */
      AalConParam atmProf;      /* ATM BB profile */
   }t;
} CcBBProfCfg;
```

CcAtmParms

The structure is:

```
typedef struct ccAtmParms    /* ATM Parameters */
{
  AmBBearCap     bBear;      /* Broadband Bearer Capability */
  CcNBearCap     nBear;      /* Narrowband Bearer Capability */
  AmAalParam     aal;        /* AAL */
  CcAtmCellRate  atmCR;      /* ATM Cell Rate */
  CcAATMCellRate addATMCR;   /* Additional ATM Cell Rate */
  CcQoS          qOfSrv;     /* Quality of Service */
  AmBHiLyrInfo   hiLyInf;    /* Broadband High Layer Information */
  CcBLoLyrInfo   loLyInf;    /* Broadband Low Layer Information */
} CcAtmParms;
```

bBear

The broadband bearer capability parameter configuration has the following format:

```
typedef struct amBBearCap  /* Broadband Bearer Capability
                              Tokens */
{
   ElmtHdr eh;             /* element header */
   TknU8   bearClass;      /* bearer class */
   TknU8   timingReq;      /* timing requirement */
   TknU8   tfcType;        /* traffic type */
   TknU8   atmTfrCap;      /* ATM transfer capability */
   TknU8   usrPlaneConCfg; /* user plane connection
                              configuration */
   TknU8   suscClip;       /* susceptability to clipping */
   TknU8 usrInfoLyr2Prot;  /* User information layer2 protocol
                            */
   TknU8   lyrIdBearer;    /* Layer ID */

} AmBBearCap;
```

The **bearClass** field can take the following values:

Value	Description
AM_BCOB_A	Bearer class A
AM_BCOB_C	Bearer class C
AM_BCOB_X	Bearer class X

As specified in recommendation Q.2723.2 / Q.2961-2, the fields in the **timingReq** and **tfcType** have been overwritten by the field, **atmTfrCap**.

The **atmTfrCap** field can take the following values:

Value	Description
AM_ATC_NRTVBR1	Non-real time VBR
AM_ATC_RTVBR1	Real time VBR
AM_ATC_NRTVBR2	Non-real time VBR
AM_ATC_CBR1	CBR
AM_ATC_CBR2	CBR
AM_ATC_CBR3	CBR
AM_ATC_CBRCLR	CBR with CLR commitment on CLP = 0+1
AM_ATC_NRTVBR3	Non-real time VBR
AM_ATC_RTVBR2	Real time VBR
AM_ATC_NRTVBR4	Non-real time VBR
AM_ATC_NRTVBRCLR	Non-real time VBR with CLR commitment on CLP = 0 + 1
AM_ATC_ABR	ABR
AM_ATC_ABTDT	ATM bloc transfer delayed transmission
AM_ATC_ABTIT	ATM bloc transfer immediate transmission
AM_ATC_RTVBRCLR	Real time VBR with CLR commitment on CLP = 0 + 1

The **usrPlaneConCfg** field can take the following values:

Value	Description
AM_CONCFG_PTPT	Point-to-point
AM_CONCFG_PTMPT	Point-to-multipoint

The **suscClip** field can take the following values:

Value	Description
AM_SUSCLP_NO	Not susceptible to clipping
AM_SUSCLP_YES	Susceptible to clipping

The **usrInfoLyr2Prot** field can take the following value:

Value	Description
AM_Q922_CORE	Core aspects of Annex A Q.922

The **lyrIdBearer** field can take the following value:

Value	Description
AM_BBC_LAYER_ID	Layer ID in the broadband bearer cap

nBear

Narrowband bearer capability parameter configuration. Currently, this parameter is not used and is reserved for future use.

```
typedef struct biNBearCapInst
{
   TknU32 q2931Head;           /* Q.2931 header */
   TknU8 infoTranCap;          /* information transfer
                                  capability */
   TknU8 codingStd;            /* coding standard */
   TknU8 infoTranRate0;        /* information transfer rate */
   TknU8 tranMode;             /* transfer mode */
   TknU8 establish;            /* establishment */
   TknU8 cfg;                  /* configuration */
   TknU8 chanStruct;           /* structure */
   TknU8 infoTranRate1;        /* information transfer rate */
   TknU8 symmetry;             /* symmetry */
   TknU8 usrInfoLyr1Prot;      /* user information layer 1
                                  protocol */
   TknU8 lyr1Ident;            /* layer 1 identity */
   TknU8 usrRate;              /* user rate */
   TknU8 negot;                /* negotiation */
   TknU8 syncAsync;            /* synchronous/asynchronous */
```

The following tokens represent a union of octets 5b.1 and 5b.2 of the narrow band bearer capability:

```
   TknU8 FlcRx_BandNeg;         /* flow control on reception or
                                   inband/outband negotiation */
   TknU8 FlcTx_Assgn;           /* flow control on transmission
                                   or assignor/assignee*/
   TknU8 NicRx_LLINeg;          /* network independent clock on
                                   reception or logical link
                                   identifier negotiation */
   TknU8 NicTx_Mode;            /* network independent clock
                                   on transmission or mode of
                                   operation */
   TknU8 Rate_MFrm;             /* intermediate rate (low bit) or
                                   Multiframe support */
   TknU8 Rate_Hdr;              /* intermediate rate (high bit)
                                   or rate adaptation Hdr/ no
                                   Headr */
   TknU8 parity;                /* parity information */
   TknU8 nmbDatBits;            /* number of data bits excluding
                                   parity bit */
   TknU8 nmbStopBits;           /* number of stop bits */
   TknU8 modemType;             /* modem type */
   TknU8 duplexMode;            /* duplex mode */
   TknU8 usrInfoLyr2Prot;       /* user information layer 2
                                   protocol */
   TknU8 lyr2Ident;             /* layer 2 identity */
   TknU8 usrInfoLyr3Prot;       /* user information layer 3
                                   protocol */
   TknU8 lyr3Ident0;            /* layer 3 identity */
} BiNBearCapInst;
```

This parameter should be compatible with ITU-T recommendation Q.2931.

The values allowed for the tokens are the same as those allowed for the same tokens in a narrowband lower layer compatibility information element.

aal

ATM adaptation layer parameter configuration. The format of the structure is:

```
typedef struct amAalParam     /* AAL Parameters Tokens */
{
   ElmtHdr eh;                /* element header */
   TknU8   aalType;           /* AAL type */
   /* Token definition for AAL-1 */
   TknU8   subTypeId;         /* Subtype Identifier */
   TknU8   subType;           /* Subtype */
   TknU8   cbrRateId;         /* CBR Rate Identifier */
   TknU8   cbrRate;           /* CBR Rate */
   TknU8   multId;            /* Multiplier Identifier */
   TknU16  multVal;           /* Multiplier value */
   TknU8   srcClkFreqMetId;   /* Source clock Frequency
                                 method identifier */
   TknU8   srcClkFreqMet;     /* Source Clock frequency
                                 method */
   TknU8   errCrMetId;        /* Error correction method
                                 identifier */
   TknU8   errCrMet;          /* Error correction method */
   TknU8   strDatTxBlkszId;   /* Structured data transfer
                                 blocksize Id. */
   TknU8   strDatTxBlksz0;    /* Structured data transfer
                                 blocksize - oct 1*/
   /* Token definition for AAL-1, except in UNI 3.0 */
   TknU8   strDatTxBlksz1;    /* Structured data transfer
                                 blocksize - oct 2*/
   /* Token definition for AAL-1 */
   TknU8   prtFillCellId;     /* Partially filled cells
                                 Identifier */
   TknU8   prtFillCellMet;    /* Partially filled cells
                                 method */
   /* Token definition for AAL-3/4 and AAL-5 */
   TknU8   fwdMaxCpcsSduSzId; /* Forward maximum CPCS-SDU
                                 size identifier */
   TknU16  fwdMaxCpcsSduSz;   /* Forward maximum CPCS-SDU
                                 size */
   TknU8   bwdMaxCpcsSduSzId; /* Backward maximum CPCS-SDU
                                 size identifier */
   TknU16  bwdMaxCpcsSduSz;   /* Backward maximum CPCS-SDU
                                 size */
   /* Token definition for AAL-3/4 only */
   TknU8   midRangeId;        /* MID Range identifier */
   TknU16  loMidRange;        /* MID Range value */
   /* Token definition for AAL-3/4 only, except in UNI 3.0 */
   TknU16  hiMidRange;        /* MID Range value */
   /* Token definition for AAL-3/4 and AAL-5 and only for
                                 UNI 3.0*/
   TknU8   modeId;            /* Mode identifier */
   TknU8   mode;              /* Mode - Streaming/Message
                                */
   /* Token definition for AAL-2 */
```

```
   TknU8   maxCpsSduSzId;     /* AAL-2 Max CPS SDU Size Id*/
   TknU8   maxCpsSduSz;       /* AAL-2 Max CPS SDU Size */
   TknU8   maxMuxchannelId;   /* AAL-2 Max AAL2 Channel Id*/
   TknU8   maxMuxchannel;     /* AAL-2 Max AAL2 Channel */

   /* Token definition for AAL2, AAL-3/4 and AAL-5 */
   TknU8   sscsTypeId;        /* SSCS Type Identifier */
   TknU8   sscsType;          /* SSCS Type */
   /* Token definition for AAL-2 (SSCS-SAR (I.366.1)) */
   TknU8   sarSscsParamId;    /* AAL-2 SSCS SAR Parameter Id */
   TknU8   errDect;           /* AAL-2 SSCS-SAR Error Det */
   TknU8   asrdDat;           /* AAL-2 SSCS-SAR Assured Data */
   TknU8   fwdMaxSsSarSduSzId; /* FWD Max SSCS-SAR SDU Sz Id*/
   TknU32  fwdMaxSsSarSduSz;  /* FWD Max SSCS-SAR SDU Size */
   TknU8   bwdMaxSsSarSduSzId; /* BWD Max SSCS-SAR SDU Sz Id*/
   TknU32  bwdMaxSsSarSduSz;  /* BWD Max SSCS-SAR SDU Size */

   /* Token definition for AAL-2 (SSCS-trunking (I.366.2)) */
   TknU8   trnkSscsParamId;   /* AAL-2 SSCS Trnkg Param Id */
   TknU8   fmd;               /* FMD */
   TknU8   cmd;               /* CMD */
   TknU8   srvcCtgry;         /* Service Categeory */
   TknU8   pcmEncdng;         /* PCM-Encoding */
   TknU8   mfR2;              /* MF-R2 */
   TknU8   mfR1;              /* MF-R1 */
   TknU8   dtmf;              /* DTMF */
   TknU8   cas;               /* CAS */
   TknU8   fax;               /* FAX */
   TknU8   mult;              /* Multiplier */
   TknU8   maxFrmMdDatUntId;  /* Max length of Frame Mode
                                 Data Unit Id*/
   TknU16  maxFrmMdDatUnt;    /* Max length of Frame Mode
                                 Data Unit */
   TknU8   prflIdentId;       /* Profile Identification Id */
   TknU8   prflSrc;           /* Profile Source*/
   TknU8   preDfnPrfl;        /* Predefined Profile */
   TknU32  ouiAal2;           /* Organisation unique
                                 identifier */

   /* Token definition for User defined AAL */
   TknU32  usrDefAalInfo;     /* User defined AAL
                                 information */
} AmAalParam;
```

The **aalType** field can take the following values:

Value	Description
AM_AALTYP_1	AAL type 1
AM_AALTYP_2	AAL type 2
AM_AALTYP_34	AAL type 3/4
AM_AALTYP_5	AAL type 5
AM_AALTYP_USR	User-defined AAL

The **subTypeId**, **cbrRateId**, **multId**, **srcClkFreqMetId**, **strDatTxBlkSzId**, and **prtFillCellId** fields can take the following values:

Value	Description
AM_AAL1_ID_STYPE	AAL subtype ID
AM_AAL1_ID_CBR	CBR rate ID
AM_AAL1_ID_MULT	Multiplier ID
AM_AAL1_ID_SCFRM	Source clock frequency recovery method ID
AM_AAL1_ID_ECM	Error correction method ID
AM_AAL1_ID_SDTB	Structured data transfer block size ID
AM_AAL1_ID_PFC	Partially filled cells ID

The **subType** field can take the following values:

Value	Description
AM_AAL1_STYPE_NULL	Null/empty
AM_AAL1_STYPE_VOICE	Voice band, based on 64 kbit/s
AM_AAL1_STYPE_SCKT	Synchronous circuit emulation
AM_AAL1_STYPE_ACKT	Asynchronous circuit emulation
AM_AAL1_STYPE_HQAUD	High quality audio
AM_AAL1_STYPE_VIDEO	Video

The **cbrRate** field can take the following values:

Value	Description
AM_AAL1_CBR_64	64 kbit/s
AM_AAL1_CBR_1544	1544 kbit/s (DS1)
AM_AAL1_CBR_6312	6312 kbit/s (DS2)
AM_AAL1_CBR_32064	32064 kbit/s
AM_AAL1_CBR_44736	44736 kbit/s (DS3)
AM_AAL1_CBR_97728	97728 kbit/s
AM_AAL1_CBR_2048	2048 kbit/s (E1)
AM_AAL1_CBR_8448	8448 kbit/s (E2)
AM_AAL1_CBR_34368	34368 kbit/s (E3)
AM_AAL1_CBR_139264	139264 kbit/s
AM_AAL1_CBR_nx64	n x 64 kbit/s
AM_AAL1_CBR_nx8	n x 8 kbit/s

The **srcClkFreqMet** field can take the following values:

Value	Description
AM_AAL1_SCFRM_NULL	NULL
AM_AAL1_SCFRM_SRTS	Synchronous residual time stamp
AM_AAL1_SCFRM_ACR	Adaptive clock recovery

The **errCrMet** field can take the following values:

Value	Description
AM_AAL1_ECM_NULL	NULL
AM_AAL1_ECM_FEC	Inter-leaved FEC
AM_AAL1_ECM_DSST	For delay-sensitive signal transport

The **strDatTxBlkSz0** field can take the following list of values.

Value	Description
AM_AAL1_SDTB_NULL	NULL
AM_AAL1_SDTB_SDT	Structured data transfer

The **fwdMaxCpcsSduSzId**, **bwdMaxCpcsSduSzId**, **midRangeId**, **modeId**, and **sscsTypeId** fields can take the following values:

Value	Description
AM_AAL5_ID_FMSDU	Forward maximum CPCS SDU size ID
AM_AAL5_ID_BMSDU	Backward maximum CPCS SDU size ID
AM_AAL5_ID_MIDRNG	Mid-range ID
AM_AAL5_ID_MODE	Mode ID
AM_AAL5_ID_SSCS	SSCS type ID

The **mode** field can take the following values:

Value	Description
AM_AAL5_MODE_MSG	Message mode
AM_AAL5_MODE_STREAM	Streaming mode

The **sscsType** fields can take the following values:

Value	Description
AM_AAL5_SSCS_NULL	NULL SSCS
AM_AAL5_SSCS_SSCOP_A	SSCOP assured mode SSCS
AM_AAL5_SSCS_SSCOP_N	SSCOP non-assured mode SSCS
AM_AAL5_SSCS_FR	Frame relay SSCS

atmCR

ATM cell rate parameter configuration.

```
typedef struct ccAtmCellRte      /* ATM Cell Rate Tokens */
{
   ElmtHdr eh;                   /* element header */
   TknU8   cellRateId1;          /* cell rate id, CLP  */
   TknU32  cellRate1;            /* cell rate,   CLP    */
   TknU8   cellRateId2;          /* cell rate id, CLP  */
   TknU32  cellRate2;            /* cell rate,   CLP    */
   TknU8   cellRateId3;          /* cell rate id, CLP  */
   TknU32  cellRate3;            /* cell rate,   CLP    */
   TknU8   cellRateId4;          /* cell rate id, CLP  */
   TknU32  cellRate4;            /* cell rate,   CLP    */
} CcAtmCellRate;
```

Note: *The fields must be encoded as stated by Q.2763.*

addATMCR

Additional ATM cell rate parameter configuration.

```
typedef struct ccAATMCellRate      /* Additional ATM Cell Rate
                                      Tokens */
{
   ElmtHdr eh;                     /* element header */
   TknU8 cellRateId1;              /* cell rate id, CLP */
   TknU32 cellRate1;               /* cell rate, CLP */
   TknU8 cellRateId2;              /* cell rate id, CLP */
   TknU32 cellRate2;               /* cell rate, CLP */
   TknU8 cellRateId3;              /* cell rate id, CLP */
   TknU32 cellRate3;               /* cell rate, CLP */
   TknU8 cellRateId4;              /* cell rate id, CLP */
   TknU32 cellRate4;               /* cell rate, CLP */
   TknU8 cellRateId5;              /* cell rate id, CLP */
   TknU32 cellRate5;               /* cell rate, CLP */
   TknU8 cellRateId6;              /* cell rate id, CLP */
   TknU32 cellRate6;               /* cell rate, CLP */
   TknU8 cellRateId7;              /* cell rate id, CLP */
   TknU32 cellRate7;               /* cell rate, CLP */
   TknU8 cellRateId8;              /* cell rate id, CLP */
   TknU32 cellRate8;               /* cell rate, CLP */
 /* Additional ATM Cell Rate Tokens for Q.2723.3 OR Q.2723.4 */
   TknU8 cellRateId9;              /* cell rate id, CLP */
   TknU32 cellRate9;               /* cell rate, CLP */
   TknU8 cellRateId10;             /* cell rate id, CLP */
   TknU32 cellRate10;              /* cell rate, CLP */
} CcAATMCellRate;
```

The **cellRateIdX** field can take the following values:

Value	Description
BI_AACR_FSCR_ID0	Forward sustainable cell rate for CLP = 0
BI_AACR_BSCR_ID0	Backward sustainable cell rate for CLP = 0
BI_AACR_FSCR_ID1	Forward sustainable cell rate for CLP = 0+1
BI_AACR_BSCR_ID1	Backward sustainable cell rate for CLP = 0+1
BI_AACR_FAMCR_ID1	Forward ABR minimum cell rate for CLP = 0+1
BI_AACR_BAMCR_ID1	Backward ABR minimum cell rate for CLP = 0+1
BI_AACR_FMBS_ID0	Forward maximum burst size for CLP = 0
BI_AACR_BMBS_ID0	Backward maximum burst size for CLP = 0 size for CLP = 0+1

BI_AACR_BMBS_ID1	Backward maximum burst size for CLP = 0+1
BI_AACR_FRMPCR_ID	Forward resource management PCR ID
BI_AACR_BRMPCR_ID	Backward resource management PCR ID
BI_AACR_RESERVED	Reserved

qOfSrv

Quality-of-Service (QoS) parameter configuration. The format of the structure is:

```
typedef struct ccQoS       /* Q.2726.3 Connection Identifier */
{
   ElmtHdr eh;             /* element header */
   TknU8 codingStd;        /* coding standard */
   TknU8 qOfServFwd;       /* quality of service forward */
   TknU8 qOfServBwd;       /* quality of service backward */
} CcQoS;
```

The **codingStd** field can take the following values:

Value	Description
AM_CSTD_INT	Other international standards
AM_CSTD_NAT	National standard
AM_CSTD_NET	Network standard

The **qOfServFwd** and **qOfServBwd** fields can take the following values:

Value	Description
BI_QOS_UNSPEC	Unspecified QoS class
BI_QOS_RESERVED	Reserved for future indications of a parametrized QoS

hiLyInf

Broadband high-layer information parameter configuration. The format is:

```
typedef struct ccBHiLyrInfo  Broadband High Layer
                             Information Tokens */
{
   ElmtHdr eh;               element header */
   TknU8   hiLyrInfoType;    high layer information
                             type */
   TknStrS hiLyrInfo;        high layer information */
} CcBHiLyrInfo;
```

The **hiLyrInfoType** field can take the following values:

Value	Description
AM_HLITYP_ISO	ISO
AM_HLITYP_USR	User-specific
AM_HLITYP_HLPROF	high-layer profile
AM_HLITYP_APPID	Vendor-specific application ID

loLyInf

Broadband lower layer information parameter configuration.

```
typedef struct ccBLoLyrInfo     /* Broadband Low Layer
                                   Information Tokens */
{
   ElmtHdr eh;                  /* element header */
   TknU32  q2931Head;           /* Q.2931 header */
   TknU8   usrInfoLyr1Prot;     /* user information layer 1
                                   protocol */
   TknU8   lyr1Id;              /* Layer 1 id */
   TknU8   usrInfoLyr2Prot;     /* user information layer 2
                                   protocol */
   TknU8   lyr2Id;              /* Layer 2 id */
   TknU8   q933Use;             /* Q.933 use */
   TknU8   lyr2OprMode;         /* Mode of operation */
   TknU8   winSize;             /* Window size */
   TknU8   usrSpecLyr2ProtInfo; /* User specified layer 2
                                   protocol info */
   TknU8   usrInfoLyr3Prot;     /* user information layer 3
                                   protocol */
   TknU8   lyr3Id;              /* Layer 3 id */
   TknU8   lyr3OprMode;         /* Mode of operation */
   TknU8   defPktSize;          /* Default packet size */
   TknU8   pktWinSize;          /* Default packet size */
   TknU8   usrSpecLyr3ProtInfo; /* User specified layer 3
                                   protocol info */
   TknU8   initProtId;          /* Initial protocol Identifier
                                   bits 8-2 */
   TknU8   snapId;              /* SNAP identifier */
   TknU32  oui;                 /* Organization unique
                                   identifier */
   TknU16  protId;              /* Protocol identifier */
} CcBLoLyrInfo;
```

The **lyr1Id** field can take the following value:

AM_L1_IDENT **layer 1 identity**

The **lyr2Id** field can take the following value.

AM_L2_IDENT **layer 2 identity**

The **lyr3Id** field can take the following value.

AM_L3_IDENT **layer 3 identity**

The **usrInfoLyr1Prot** field can take the following values:

Value	Description
AM_UIL1_CCITTV11	CCITT standardized rate adaptation V.110/X.30
AM_UIL1_G711ULAW	Recommendation G.711 U-Law
AM_UIL1_G711ALAW	Recommendation G.711 A-Law
AM_UIL1_G721ADCPM	Recommendation G.721 32 kbit/s ADCPM
AM_UIL1_G722G725	Recommendation G.722 and G.725 - 7kHz audio
AM_UIL1_H261	Recommendation H.261 - 384 kbit/s video
AM_UIL1_NONCCITT	Non-CCITT standardized rate adaptation
AM_UIL1_CCITTV120	CCITT standardized rate adaptation V.120
AM_UIL1_CCITTX31	CCITT standardized rate adaptation X.31 HDLC

The **usrInfoLyr2Prot** field can take the following values:

Value	Description
AM_UIL2_BASIC	Basic mode—ISO 1745
AM_UIL2_Q921	CCITT recommendation Q.921
AM_UIL2_X25SLP	CCITT recommendation X.25, single link
AM_UIL2_X25MLP	CCITT recommendation X.25, multi link
AM_UIL2_T71	Extended LAPB for half duplex
AM_UIL2_HDLCARM	HDLC ARM—ISO 4335
AM_UIL2_HDLCNRM	HDLC NRM—ISO 4335
AM_UIL2_HDLCABM	HDLC ABM—ISO 4335
AM_UIL2_LANLLC	LAN LLC—ISO 8802/2
AM_UIL2_X75SLP	CCITT recommendation X.75, single link
AM_UIL2_Q922	CCITT recommendation Q.922
AM_UIL2_USRSPEC	CCITT user-specified
AM_UIL2_T90	CCITT T.90

The **lyr3OprMode** and **lyr2OprMode** fields can take the following values:

Value	Description
AM_LOLYR_OPR_NORM	Normal mode of operation
AM_LOLYR_OPR_EXT	Extended mode of operation

The **usrInfoLyr3Prot** field can take the following values:

Value	Description
AM_UIL3_Q931	CCITT recommendation Q.931
AM_UIL3_T90	CCITT T.90
AM_UIL3_X25PLP	CCITT recommendation X.25, packet layer
AM_UIL3_ISO8208	ISO 8208
AM_UIL3_ISO8348	ISO 8348
AM_UIL3_ISO8473	ISO 8473
AM_UIL3_T70	CCITT recommendation T.70
AM_UIL3_USRSPEC	CCITT user-specified

Abbreviations

The following abbreviations are used in this document:

Abbreviation	Description
ANSI	American National Standards Institute: A U.S. standards body.
ATM	Asynchronous Transfer Mode: A transfer mode in which the information is organized into cells. It is asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic.
BCR	Backward Cell Rate: Cell rate in the backward direction of a call.
B-ICI	B-ISDN Inter-Carrier Interface: An ATM Forum-defined specification for the interface between public ATM networks to support user services across multiple public carriers.
B-ISUP	Broadband ISDN User's Part: An SS7 protocol that defines the signalling messages to control connections and services.
BRI	Basic Rate Interface
CAC	Connection Admission Control: Connection Admission Control is defined as the set of actions taken by the network during the call setup phase (or during call re-negotiation phase) in order to determine whether a connection request can be accepted or should be rejected (or whether a request for re-allocation can be accommodated).
CBR	Constant Bit Rate
CIC	Circuit Identification Code
DPC	Destination Point Code
DSS1	Digital Subscriber Signalling System #1: N-ISDN UNI Signalling
DSS2	Digital Subscriber Signalling System #2: B-ISDN UNI Signalling
FCR	Forward Cell Rate: Cell rate in the forward direction of a call.
GAP	GCC Audit Procedure
GCC	Generic Call Control
ICC	Interworking Call Control: A call control used for interworking between two or more different protocols.
ISDN	Integrated Services Digital Network. Communication protocol that permits telephone networks to carry data, voice, and other source traffic.

Abbreviation	Description
ISUP	ISDN User Part: An SS7 protocol that provides the signalling functions required to support basic bearer services and supplementary services for voice and non-voice applications in an ISDN.
ITU-T	International Telecommunications Union Telecommunications: ITU-T is an international body of member countries whose task is to define recommendations and standards relating to the international telecommunications industry. It was previously known as CCITT.
NFAS	Non-Facility Associated Signalling
N-ISDN	Narrowband Integrated Services Digital Network: Services include basic rate interface (2B+D or BRI) and primary rate interface (30B+D - Europe and 23B+D - North America or PRI).
NNI	Network Node Interface: An interface between switches defined as the interface between two network nodes.
OAP	One-time audit procedure
PAP	Periodic audit procedure
PNNI	Private Network - Network Interface. A routing information protocol that enables extremely scalable, full function, dynamic multi-vendor ATM switches integrated in the same network.
PRI	Primary Rate Interface: An ISDN standard for the provisioning of 1.544 Mbit/s (DS1 - North America, Japan) or 2.048 Mbit/s (E1 - Europe) ISDN services. DS1 is 23 "B" channels of 64 kbit/s each and one signalling "D" channel of 64 kbit/s; E1 is 30 "B" channels of 64 kbit/s each and one signalling "D" channel of 64 kbit/s.
PSIF	Protocol-Specific Interface Function: Interface between the Generic Call Control and the outgoing or incoming protocol.
RM	Resource Management: The management of resources in a network.
RT	Router: Entity that finds the destination interface for an incoming call.
SAP	Service Access Point
SFM	Switching Fabric Manager: Interface to the switching fabric.
SM	Stack Manager
SS	System Services: Interface with the operating system.
SS7	Signalling System 7: A family of signalling protocols originating from narrowband telephony. They are used to set up, manage, and tear down connections, as well as to exchange non-connection associated information.
TAPA	Trillium Advanced Portability Architecture: A set of primitives and guidelines for a Trillium product.

Abbreviation	Description
VBR	Variable Bit Rate
VCCI	Virtual Channel Connection ID
VCI	Virtual Channel ID: A unique numerical tag as defined by a 16-bit field in the ATM cell header that identifies a virtual channel over which the cell travels.
VPI	Virtual Path ID: An 8-bit field in the ATM cell header that indicates the virtual path over which the cell should be routed.
XM	Connection Manager

References

Refer to the following documents for additional information.

AMT Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1100003).

BIT Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1100009).

CCT Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1100023).

INT Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1100018).

Interworking Between ISUP (ANSI) and B-ISUP (ITU-T), Trillium Digital Systems, Inc. (p / n 1112001).

Interworking Between ISUP (ITU) and PNNI (ATM Forum), Trillium Digital Systems, Inc. (p / n 1112002).

Interworking Call Control Functional Specification, Trillium Digital Systems, Inc.(p / n 1091134).

Interworking Call Control Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1100022).

Interworking Call Control Portation Guide, Trillium Digital Systems, Inc. (p / n 1093134).

Interworking Call Control Service Definition, Trillium Digital Systems, Inc. (p / n 1092134).

Interworking Call Control Software Test Sample, Trillium Digital Systems, Inc. (p / n 1094134).

Interworking Call Control Training Course, Trillium Digital Systems, Inc. (p / n 1095134).

Open Systems Interconnection - Basic Reference Model, ISO (IS 7498).

Open Systems Interconnection - Basic Reference Model Addendum 1: Connectionless Data Transmission, ISO (IS 7498 DAD 1).

PSIF - B-ISUP Service Definition, Trillium Digital Systems, Inc. (p / n 1092143).

PSIF - ISUP Service Definition, Trillium Digital Systems, Inc. (p / n 1092141).

PSIF - Q.930/Q.931 Service Definition, Trillium Digital Systems, Inc. (p / n 1092140).

PSIF - Q.93B Service Definition, Trillium Digital Systems, Inc. (p / n 1092142).

Q.699—Interworking Between ISDN Access and Non-ISDN Access Over ISDN User Part of Signalling System No. 7, ITU-T.

Q.2660—Interworking Between Signalling System No. 7 - Broadband ISDN User Part (B-ISUP) and Narrowband ISDN User Part (N-ISUP), ITU-T.

Reference Model of Open Systems Interconnection for CCITT Applications, CCITT (X. 200).

SIT Interface Service Definition, Trillium Digital Systems, Inc.

System Services Interface Service Definition, Trillium Digital Systems, Inc. (p / n 1111001).

T1.656—Interworking between Signalling System No. 7-Broadband ISDN User Part (B-ISUP) and Narrowband ISDN User Part (N-ISUP), ANSI.

AF-VTOA-0089.000—Voice and Telephony Over ATM - ATM Trunking using AAL1 for Narrowband Services, version 1.0.

AF-VTOA-0113.000—ATM Trunking Using AAL2 for Narrowband Services.