INTERNATIONAL STANDARD

ISO/IEC 4396-1

First edition
2023-12

Telecommunications and information exchange between systems — Recursive inter-network architecture —

Part 1: Reference model

Télécommunications et échange d'information entre systèmes — Architecture récursive inter-réseaux —

Partie 1: Modèle de référence

Reference number
ISO/IEC 4396-1:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6 *Telecommunications and information exchange between systems*.

A list of all parts in the ISO/IEC 4396 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

The purpose of this document is to provide a reference model for the concepts in Recursive Inter-Network Architecture (RINA)[1],[2]. This document provides the fundamental definitions for the ISO/IEC 4396 series. It defines an architecture.

This document is the high-level description of the concepts, the elements and how they work. This is a top-level specification, not a tutorial. Other more detailed specifications will draw on the ISO/IEC 4396 series as their starting point.

Telecommunications and information exchange between systems — Recursive inter-network architecture —

Part 1: Reference model

1 Scope

This document provides the reference model for the Recursive Inter-Network Architecture (RINA). It describes:

— the basic concepts of distributed systems and distributed applications;

— distributed management systems (DMSs);

— the fundamental structure of distributed Inter-Process Communications;

— the Distributed Inter-Process Facility (DIF) operations.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
address
identifier that is a synonym for the fully qualified IPC-Process-Instance-name, which is a member of a *distributed-IPC-facility (DIF)* (3.32)

Note 1 to entry: An address is only unambiguous within the DIF (and is assigned by the DIF). This identifier can be assigned to facilitate its usefulness to the operation of the DIF, i.e. location-dependent.

3.2
application connection
shared state maintained by two communicating peer *application entities (AEs)* (3.3)

Note 1 to entry: Application connections go initially through an establishment phase, in which enough data is exchanged to establish a shared understanding between both AEs, to later proceed to the data transfer phase, in which both AEs exchange data. In Recursive Inter-Network Architecture (RINA), the establishment phase is handled by *common application connection establishment procedure (CACEP)* (3.17), while the data transfer phase is the responsibility of *common distributed application protocol (CDAP)* (3.18).

3.3
application-entity
AE
task within an application process that is directly involved with exchanging application information with other *application processes (APs)* (3.9)

3.4
application entity instance
AE-instance
AEI
instantiation of an *application entity (AE)* (3.3) within an *application process (AP)* (3.9)

3.5
application entity instance identifier
AE-instance-id
AEI-id
identifier which is unambiguous within the *application entity (AE)* (3.3)

Note 1 to entry: The AE-instance-id may be ambiguous within the *application process (AP) name space* (3.13) unless qualified by the application process name, *application process instance id* (3.11), and the *application entity (AE) name* (3.6).

3.6
application entity name
AE name
identifier from the *application entity (AE) name space* (3.8) which is unambiguous within the *scope* (3.55) of the *application process (AP)* (3.9)

Note 1 to entry: An AE name when concatenated with an *AP name* (3.12) is unambiguous within the *AP name space* (3.13), as is an AE name concatenated with an AP name and an *AP instance id* (3.11).

3.7
application programming interface primitive
API primitive
library or system call used by an application to invoke functions, in particular *inter process communication (IPC)* (3.39) functions, such as requesting the allocation of IPC resources

3.8
application entity name space
AE name space
set of strings which may be assigned to *application entities (AEs)* (3.3) of a given *application process (AP)* (3.9) and used to reference them by other applications in the same naming domain

3.9
application- process
AP
software program in a *processing system* (3.46) intended to accomplish some purpose

Note 1 to entry: An application process contains one or more tasks or *application entities (AEs)* (3.3) as well as functions for managing the resources (e.g. processor, storage, and IPC) allocated to this AP.

Note 2 to entry: Tasks are also application processes.

3.10
application process instance
AP instance
instantiation of an *application process (AP)* (3.9) on an operating system

3.11
application process instance id
AP instance id
identifier that is unambiguous within the *application process (AP)* (3.9) and is bound to an *AP instance* (3.10) in order to distinguish among multiple AP instances

Note 1 to entry: An AP instance id concatenated with an *AP name* (3.12) is unambiguous in the *AP name space* (3.13).

3.12
application -process -name
AP -name
string assigned to a single *application process (AP)* (3.9) from an *AP name space* (3.13)

Note 1 to entry: An AP name is not assigned to any other AP while bound to the one to which it has been assigned.

3.13
application process name space
AP name space
set of strings which may be assigned to *application processes (APs)* (3.9) and used to reference them by other APs in the same naming domain

3.14
application protocol
protocol used between two *application entities (AEs)* (3.3) to perform operations external to the *protocol machine (PM)* (3.50) itself

Note 1 to entry: The distinguishing characteristic of application protocols is that they modify states external to the protocol.

3.15
assignment
operation that allocates a name in a *name space* (3.43), essentially marking it as being in use

Note 1 to entry: Assignment makes names available to be bound. This allows certain portions of a name space to be "reserved" and not be available for *binding* (3.16). The corresponding reverse operation, *de-assignment* (3.25), removes it from use.

3.16
binding
function, $\mathbf{F_M}$,$\mathbf{N_S}$, that defines the mapping of a subset of elements of {**NS**} to elements of {**M**}

Note 1 to entry: This function is one-to-one and into. The operation, *binding*, binds a name to an object.

Note 2 to entry: Once bound, any reference to the name locates or accesses the object.

3.17
common application connection establishment procedure
CACEP
procedure to authenticate flow participants and initialize the application naming and protocol information

Note 1 to entry: CACEP naming and protocol information relates to the *application protocol* (3.14) that will be used by applications to exchange information (e.g. abstract and encoding rules, object model versions). In case of Recursive Inter-Network Architecture (RINA), the application protocol is *common distributed application protocol (CDAP)* (3.18).

3.18
common distributed application protocol
CDAP
application protocol (3.14) component of a *distributed application facility (DAF)* (3.27) used to construct arbitrary distributed applications

Note 1 to entry: CDAP enables distributed applications to deal with communications at an object level, rather than forcing applications to explicitly deal with serialization and input/output operations. CDAP provides a straightforward and unifying approach to sharing data over a network without having to create specialized protocols.

Note 2 to entry: *Distributed IPC facility (DIF)* (3.32) is an example of a *distributed application facility (DAF)* (3.27).

3.19
computing system
collection of all *processing systems* (3.46) (some specialized) in the same management domain

Note 1 to entry: There are no restrictions on the connectivity of computing systems.

3.20
connection
shared state between *error and flow control protocol machine EFCP PMs* (3.34)

3.21
connection-endpoint-identifier
CEP-id
identifier that is unambiguous within the *scope* (3.55) of an *interprocess communication (IPC)* process which identifies an *error and flow control protocol machine EFCP PM* (3.34) instance

3.22
connection-identifier
identifier internal to the *distributed IPC facility (DIF)* (3.32) that are unambiguous within the scope of communicating *error and flow control protocol machine EFCP PMs* (3.34) from that DIF

Note 1 to entry: The connection identifier is formed by the concatenation of the source and destination connection establishment procedure (CEP)-ids to identify the two directions of the connection.

3.23
data transfer control procedure
DTCP
half of an *error and flow control protocol (EFCP)* (3.33) that performs loosely bound (feedback) mechanisms, such as retransmission and flow control

Note 1 to entry: The DTCP protocol machine (PM) maintains state, which can be discarded after long periods of no traffic.

Note 2 to entry: One instance of a DTCP is created for each connection of a flow.

Note 3 to entry: All connections in Recursive Inter-Network Architecture (RINA) have flow control. Connections without flow control are denial of service attack vector.

3.24
data transfer procedure
DTP
half of an *error and flow control protocol (EFCP)* (3.33) that performs tightly bound mechanisms, such as ordering and fragmentation/reassembly

Note 1 to entry: One instance of a DTP protocol machine (PM) is created for each connection allocated.

3.25
de-assignment
operation that deallocates a name in a *name space* (3.43), removing it from use

3.26
delimiting
operation to delineate the beginning and end of a *service-data-unit (SDU)* (3.56) and package it into the user-data field of a *protocol-data-unit (PDU)* (3.45)

Note 1 to entry: Delimiting is usually the first operation performed by the *distributed IPC facility (DIF)* (3.32) when an SDU is submitted.

Note 2 to entry: Delimiting enables the DIF to deliver the SDU as a unit of data to its recipient intact.

3.27
distributed application facility
DAF
distributed application
collection of *application processes (APs)* (3.9) in *processing systems* (3.46) that exchange information using *interprocess communication (IPC)* (3.39) and maintain shared state to cooperate in performing some task or function

Note 1 to entry: There are at least two APs and at least one processing system in each DAF.

Note 2 to entry: The DAF forms a black box to the members of the DAF who may be executing on one or more processing systems.

Note 3 to entry: In some DAF, all members of the DAF will be the same, i.e. a homogeneous DAF, while in others they may be different, i.e. a heterogeneous DAF.

3.28
distributed application name
whatevercast name (3.61) for the set of *application processes (APs)* (3.9) comprising a distributed application depending on the operation

Note 1 to entry: A whatevercast name is generally taken from the same *name space* (3.43) as the APs and is used to identify a distributed application. An important type of distributed application is a *distributed IPC facility (DIF)* (3.32), i.e. the set of cooperating *interprocess communication (IPC)* (3.39) processes.

3.29
distributed application process
DAP
application process (AP) (3.9) that is a member of a *distributed application facility (DAF)* (3.27)

3.30
distributed application process name
DAP name
synonym for an *application process (AP)* (3.9) name

3.31
distributed application process synonym
DAP synonym
synonym for a *distributed application process (DAP)* (3.30) that is a member of a specific *distributed application facility DAF* (3.27) and is only unambiguous within the DAF (and is assigned by the DAF)

Note 1 to entry: The names may be structured to facilitate their use within the DAF.

3.32
distributed IPC facility
DIF
layer
collection of *application process (AP) instances* (3.10) that are cooperating to provide *interprocess communication (IPC)* (3.9)

Note 1 to entry: A DIF is a *distributed application facility (DAF)* (3.27) that does IPC.

Note 2 to entry: The DIF provides IPC services to AP instances of a DAF or IPC process instances of other DIFs via a set of application programming interface (API) primitives that are used to exchange information with the IPC process instances' peer.

3.33
error and flow control protocol
EFCP
data transfer protocol used to maintain an instance of *interprocess communication (IPC)* (3.39) within a *distributed IPF facility (DIF)* (3.32).

Note 1 to entry: The functions of this protocol can be used to provide reliability, order and flow control as required as determined by policy.

3.34
EFCP protocol machine
EFCP PM
instance of the *error and flow control protocol (EFCP)* (3.33) for a single connection

Note 1 to entry: An EFCP PM consists of two state machines loosely coupled through a single state vector: one that performs the *data transfer procedure (DTP)* (3.24) *protocol machine (PM)* (3.50) and the other that performs the *data transfer control procedure (DTCP)* (3.23) *protocol machine (PM)* (3.50).

3.35
flow
binding (3.16) of a connection to source and destination ports

3.36
flow allocator
FA
task that handles requests to allocate a *flow* (3.35)

3.37
flow allocator instance
FA-instance
FAI
instance created for each allocation request to manage the flow for its lifetime

Note 1 to entry: The flow allocator instance will translate the (quality of service) QoS requested by the Application-Process into specific policies and find the destination Application and determine if the allocation can be honoured. The FAI-identifier or port–id is returned to the application as a handle for referencing the allocation.

3.38
IPC process
IPCP
application process (AP) (3.9) whose primary purpose is managing *inter process communication (IPC)* (3.39)

3.39
inter process communication
IPC
service that allows two or more *application process (AP) instances* (3.10) to exchange data

3.40
IPC resource manager
IRM
component of a *distributed application facility (DAF)* (3.27) that manages its use of IPC

3.41
multiplexing task
MT
task within the *interprocess communication (IPC)* (3.39) process that performs multiplexing onto (N-1)-ports

Note 1 to entry: There is potentially one MT for each (N-1)-port.

3.42
name
unique string, **N**, in some alphabet, **A**, that, when seen within a *name space (NS)* (3.43) unambiguously denotes an object or denotes a statement in some language, **L**, constructed using the alphabet, **A**

3.43
name space
NS
set of names, {**N**}, from which all names for a given collection of objects are taken

Note 1 to entry: A name from a given name space may be bound to one and only one object at a time.

3.44
port
binding (3.16) of an (N)- error and flow control protocol *(EFCP) protocol machine (PM)* (3.34) instance to either an *application entity (AE) instance* (3.4) (if a *distributed application facility (DAF)* (3.27)) or an (N)-IPC process instance [if a *distributed IPC facility (DIF)* (3.32)] in the layer above

3.45
port id
identifier that is unambiguous within the *scope* (3.55) of the *application process (AP) instance* (3.10) and a *distributed IPC facility (DIF)* (3.32) interprocess communication process (IPCP) instance that is used to distinguish an instance of communication, i.e. a specific IPC allocation

Note 1 to entry: A port-id is a flow-allocator-instance-identifier (FAI-id) unique within the *flow allocator* (3.36) task, which identifies a flow-allocator-instance (FAI). FAI-id or port-id is returned to requesting applications as a handle to refer to this instance of IPC. The FAI maintains a mapping between the Port-id and the Connection-End-Point-id.

3.46
processing system
hardware and software that are capable of executing programs instantiated as *application processes (APs)* (3.9) that can coordinate using the equivalent of a “test and set” instruction, i.e. the tasks can all atomically reference the same memory

3.47
protocol
syntax and procedures of *protocol data units (PDUs)* (3.49) that specify the behaviour of two *protocol machines (PMs)* (3.50) cooperating for the purpose of maintaining shared state

3.48
protocol control information
PCI
that part of a *protocol data unit (PDU)* (3.49) that is interpreted by the *protocol machine (PM)* (3.50) in order to maintain the shared state of the protocol

3.49
protocol data unit
PDU
unit of data exchanged by *protocol machines (PMs)* (3.50) consisting of *protocol control information (PCI)* (3.48) and user data

3.50
protocol machine
PM
finite state machine that implements a protocol for maintaining a shared state

Note 1 to entry: A PM accepts input from its user and exchanges *protocol data units (PDUs)* (3.49) with a peer by transferring *service data units (SDUs)* (3.56) to a supporting service.

Note 2 to entry: The intent is to maintain the shared state with a corresponding PM, usually in another *processing system* (3.46).

3.51
relaying task
RT
task within an *interprocess communication (IPC)* (3.39) process that performs relaying of *protocol data units (PDUs)* (3.49)

Note 1 to entry: There is one RT in each IPC process.

3.52
resource allocator
component of the *interprocess communication (IPC)* (3.39) process that manages resource allocation and monitors the resources in the *distributed IPC facility (DIF)* (3.32) by sharing information with other DIF IPC processes and the performance of supporting DIFs

3.53
resource information base
RIB
logical representation of the local repository of the objects for the *distributed application facility DAF* (3.27)

Note 1 to entry: Each member of the DAF maintains a RIB.

Note 2 to entry: A distributed application may define a RIB to be its local representation of its view of the distributed application.

Note 3 to entry: From the point of view of an OS model, this is storage.

3.54
RIB daemon
common component of an *application entity (AE)* (3.3)

Note 1 to entry: a common component of the AE is a task in a local *application process (AP)* (3.9) may request the *resource information base (RIB)* (3.53) daemon to provide information from other members of the *distributed application facility (DAF)* (3.27) either on demand, on an event, or periodically.

Note 2 to entry: The RIB daemon can optimize AP requests for information.

Note 3 to entry: From the point of view of an OS model, this can be considered storage management.

3.55
scope
function, **MNS**, which defines the class of objects, {**M**}, that can be named with elements of {**NS**}

Note 1 to entry: This is sometimes referred to as the scope of the name space.

Note 2 to entry: This can also refer to actual objects or the potential for objects to be created.

3.56
service data unit
SDU
contiguous amount of data passed by an application process or IPC-Process in an IPC API primitive with integrity preserved during delivery to a corresponding application process protocol machine (AP PM)

3.57
service data unit protection
SDU protection
interprocess communication (IPC) (3.39) function that protects *service data units (SDUs)* (3.56) from being given to the (N-1)- *distributed IPC facility (DIF)* (3.32) for delivery

Note 1 to entry: SDU protection may include (but is not limited to) integrity preservation, encryption, time-to-live functions and confidentiality.

3.58
synonym
multiple names assigned to the same object

3.59
unbinding
breaking the *binding* (3.16) between a name and an object

Note 1 to entry: Once unbound, any reference will not access any object.

3.60
user data
portion of a *protocol data unit (PDU)* (3.49) that is not interpretable by the *protocol machine (PM)* (3.50) and is delivered transparently to its destination *application process (AP)* (3.9) as a *service data unit (SDU)* (3.56)

3.61
whatevercast name
identifier that is the name of a set, usually taken from a *name space* (3.43) from the same name space as the members of the set with an associated rule

Note 1 to entry: When the name is referenced, the rule is used to select names from the set.

Note 2 to entry: Traditional multicast and anycast names are specific forms of whatevercast.

Note 3 to entry: A unicast name or *address* (3.1) can be considered as a degenerate case of a whatevercast name where there is only one element in the set.

Note 4 to entry: No assumptions are made about the static or dynamic membership of the set.

4 Symbols and abbreviated terms

AE	Application-Entity
AEI	Application-Entity-Instance
AEI-id	Application-Entity-Instance-Identifier
AP	Application-Process
API	Application Programming Interface
CACEP	Common Application Connection Establishment Procedure
CDAP	Common Distributed-Application Protocol
CEP-id	Connection-Endpoint-Identifier
DTCP	Data Transfer Control Procedures
DTP	Data Transfer Procedures

DA	Distributed-Application
DAF	Distributed-Application-Facility
DAN	Distributed-Application-Name
DAP	Distributed-Application-Process
DIF	Distributed-IPC-Facility
DMS	Distributed Management System
DTP	Data Transfer Procedure
DTCP	Data Transfer Control Procedure
EFCP	Error and Flow Control Protocol
FA	Flow-Allocator
FAI	Flow-Allocator-Instance
IPCP	IPC-Process
IPC	InterProcess Communication
IRM	IPC Resource Manager
MT	Multiplexing Task
NS	Name Space
PCI	Protocol-Control-Information
PDU	Protocol-Data-Unit
PM	Protocol Machine
QoS	Quality of Service
RA	Resource Allocator
RIB	Resource Information Base
RINA	Recursive Inter-Network Architecture
SDU	Service-Data-Unit

5 Basic concepts of distributed applications

5.1 Introduction to distributed applications

A Distributed-Application is a set of two or more Application-Processes that can cooperate to perform some function. A set of AP Instances, a Distributed-Application Facility (DAF) and the members of the DAF are called Distributed-Application-Process Instances (DAP-Is).

There are several kinds of DAFs that are useful in providing infrastructure for DAF operations. The most important of these are DAFs that provide Inter-Process Communication. These DAFs are called Distributed IPC Facilities (DIFs) and are discussed in much greater detail in Clause 6. DIFs are necessary when IPC using shared memory synchronization is not possible. Other DAFs of interest are Management-DAFs for managing both DIFs and DAFs (see 5.2, 5.3 and Annex A).

Application processes or DAPs in the same processing systems may use different DIFs of the same or different rank, with maybe different scopes. All communicating Application-Process Instances are, by definition, part of a DAF. DAFs operate over any (N)-DIF that they have access to and which has sufficient scope to communicate with the members of the DAF.

5.2 Naming concepts for Distributed-Application Facilities (DAFs)

A DAF is a set of (two or more) cooperating Application-Process Instances. A DAF is assigned a distributed application-name (DAN). There may also be distributed application instance identifiers (DAII) if there are multiple instances of the same DAF.

The DAN is a whatevercast name of a set that contains the DAP-Names of all of the DAPs in the DAF and a rule to select a subset when the name is referenced. A DAF may have more than one whatevercast name that has different access control and different rules. When an Application-Process joins a DAF, it may also be assigned a synonym (or alias), which is only unambiguous within the scope of the DAF and may be structured to facilitate its use within the DAF. More than one set of synonyms may be assigned to the DAP-Is. Different sets of synonyms would be used independently within the DAF for different purposes. An Application-Process may have access to several DIFs, but not all DIFs in a processing system.

When a DAF is created, at least two whatevercast names for the set of all members of the DAF are created:

a) one has the rule to return all members of the set, i.e. a multicast name;

b) one resolves to a single member.

This latter whatevercast name has a rule that may be the “nearest” member (for some concept of nearest) or a designated member. This name should have a supporting DIF in common with the DAP-I resolving the name.

The RIB Daemon manages the mapping between namespaces this level presents to its users, among any internal naming, and between any internal namespaces and the namespaces provided by the supporting level, e.g. it translates the logical “where” of this layer into the “physical” “where” of the layer below. The scope of the namespaces supported by a DAF to its users may be greater than the scope of any one DAF.

5.3 Elements of a Distributed-Application Facility (DAF)

5.3.1 Overview

5.3.1.1 General

DAFs have common elements. Application processes have infrastructure to manage their local resources: task scheduling, memory management, and IPC. A DAF has the same four components, but with the difference that they are distributed.

Potentially each DAP has tasks to perform the same three management components for the DAF. They coordinate with their peers to support the operation of the cooperating tasks that make up the DAF. In the case of the DAF, the DAF Management task may have a range of functionality from merely providing access to management data to a distributed management functionality.

— Task scheduling (Resource Allocation) may send work to different members for execution, either because they have unique capabilities, e.g. print a document, or for load balancing. In a DIF, routing and resource allocation are Task Scheduling policies.

— Memory management (RIB Daemon) has the dual role of ensuring information is available to tasks in a timely manner and ensuring that the distributed data of the DAF is managed. The former may entail periodic retrieval and/or distribution of information, either periodically or on specific events and maintaining a given level of replication, synchronization, etc. In networking terms, this combines what has traditionally been known as routing update and event management.

— IPC Management manages the underlying communication resources used by the DAP and its tasks, ensuring that tasks have flows with given quality of service. IPC Management balances the communication requirements of the tasks with the efficiency requirements of the DAF.

— DAF Management is concerned with DAF enrollment and overall management, similar to the management agent.

5.3.1.2 Task scheduling

This DAF infrastructure task is responsible for coordinating processing load among the members of the DAF. For DIFs, the function is routing and resource allocation.

5.3.1.3 Memory Management (RIB Daemon)

The RIB is the logical view of the local information relative to the function of the DAF. For a DAF, Resource Information Base (RIB) is the logical representation of the local repository of the objects. Each member of the DAF maintains a RIB. A Distributed-Application may define a RIB to be its local representation of its view of the distributed application. From the point of view of the OS model, this is storage.

A common component is a RIB Daemon, from which other tasks may request information held by other members of the DAF either on demand, on an event, or periodically. The RIB Daemon can then optimize these requests for information.

The term "RIB" is used very loosely and need not imply a complete Object Manager or database system. It is just interpreted as a generic term for DAF-related storage without constraining the range of implementation.

The role of the RIB Daemon, as it is with Memory Management, is to make data in the application's name space available as efficiently as possible and to delay the tasks using the data as little as possible. A RIB Daemon manages the maintenance of information and its veracity for the DAF, i.e. the degree of veracity, the aging of replicated data, ACID properties if any.

The tasks of DAF members should be able to register subscriptions with the RIB Daemon to have information collected from or distributed to a set of members of the DAF on a periodic or event driven basis.

Consideration in the design of a DAF should be given to making the RIB Daemon the only generator of application SDUs to the underlying DIF. This not only allows better optimization and control, but facilitates the shift from an IPC model to a programming language model. Hence, the work of the DAF can be expressed entirely in terms of operations on the RIB.

5.3.1.4 IPC management

The IPC Management task manages the use of supporting IPC resources, i.e. an underlying DIF, by the tasks of the DAP. The primary functions of IPC Management are:

a) SDU Protection, protecting the integrity and confidentiality of communication, appears in the DAF so that application data can be protected from compromise by the top-level DIF. This module provides any required protection for the SDUs this distributed application may pass to an IPC facility, distributed or otherwise. Any data corruption protection over the data and PCI including life-time guards (e.g. TTL, hop count) and/or encryption are performed here. SDU Protection may be different on each allocated flow. It is strongly recommended that all DAF flows have confidentiality and integrity protection.

b) Multiplexing SDUs from multiple DAP tasks onto one or more supporting flows, or more than one supporting flows onto a single stream to a DAP task. The DAF may have policies for multiplexing multiple allocation requests onto the supporting DIF, maintaining pools of flows, etc. It is also possible for the DAF to support reverse multiplexing, i.e. combining several incoming flows into a

single flow. This may require a policy for marks to be inserted in the data stream for synchronizing across the flows.

c) IPC Resource Manager (IRM) provides the policy coordination among the elements of IPC Management of a DAF. The primary role of the IRM is managing the use of supporting DIFs and in some cases, participate in creating new DIFs (for more detail, see Clause 6).

d) DIF Allocator Agent. When IPC Management receives an Allocate Request for IPC resources, if the IRM fails to find the requested application supported by any of the DIFs available to it, it will consult the DIF Allocator to find a DIF that does support the requested application. The result of consulting the DIF Allocator will be to join a DIF. Whether this requires the creation of a new DIF will be determined by the DIF Allocator interacting with the Management Systems responsible for the resources required to create new IPC-Processes.

Flows managed by IPC Management are with specified AP-names. These AP-names may be primitive, or names of sets, i.e. whatevercast names.

Each of these tasks coordinates with its peers in the DAF. When a new member joins a DAF, it is made a member of a whatevercast set with a "for all" rule. There is at least a multicast group consisting of all members of the DAF. The number of flows required for this coordination depends on the DAF, i.e. all coordination can be over a single flow, or each can have a dedicated flow, or something in between. Flows can exist only with "nearest" neighbours, parent/child structures, fully-connected, etc. If a DAF uses multiple management flows internally, it is more likely that they will be allocated to better optimize performance and load, rather than functional differences.

5.3.2 DAF security principles

The primary task of security for DAFs is to protect itself from malicious attacks or the collection of sensitive information about its operation. There are five security services that may be supported by a DAF: Authentication, Access Control, Confidentiality, Integrity, and Non-Repudiation.

All members of a DAF should be authenticated as legitimate members of the DAF (this is part of the Enrollment Phase). While Access Control most commonly appears in the (N)-DIFs directly supporting Applications, AE, IRMs may be used for the same purpose within a DAF. When a (N)-PDU is passed as an (N-1)-SDU, and if confidentiality measures are in place, the (N-1)-SDU cannot be interpreted by the (N-1)-IPC-Process, i.e. no elements of the (N)-SDU are clear-text. No identifier should be used for more than one purpose. Multi-level security is a DAF issue. Any method to support multiple levels in the DIFs below violates security by distinguishing multiple levels of security. A DAF is a securable container. The security level of the DAF depends on the policies used.

5.3.3 Common Distributed-Application Protocol (CDAP)

5.3.3.1 General

The Common Distributed-Application Protocol (CDAP) is a platform for building all distributed applications[4]. The use of CDAP is to make a clean transition from the IPC model of the DIF to the programming model of DAFs. While only a single application protocol is required, the CDAP protocol is constructed from modules (see ISO/IEC 15954). The modular approach is taken for two reasons:

— to provide flexibility in accommodating legacy protocols;

— to accommodate better "language support" in the future.

5.3.3.2 Basic form of CDAP

The basic form of CDAP consists of three components:

— the common application connection establishment procedure (CACEP)[5];

— the authentication module;

— the CDAP module.

The only operations that applications can perform are create/delete, read/write, and start/stop (execute/suspend) on objects. It is the collection of objects associated with a dialogue (connection) that distinguish application entities (AEs).

These three elements are combined along with access control constraints on a set of objects to define an AE. An application process may have one or more AEs.

The only part of CDAP that can be modified is the Authentication Module. The specifics of the application are embodied in the objects associated with CDAP. The objects are associated with, not bound exclusively to, the AE. The set of objects associated with an “application entity” is in essence, defined by the access control domain.

The method for defining objects is outside the scope of this document. The protocol is intended to be independent of specific object paradigms. These specifications are a platform because alone they do not define a complete application. The CACEP module is based on a simplified version of the OSI ACSE protocol[6]. ACSE was chosen for three reasons:

— it already exists;

— it provides all of the necessary functions and no more;

— it was designed to be used recursively.

ACSE provides the basic requirements for exchanging application naming and context information and provides for an authentication module to be included. This protocol was designed to provide a common means to create an application connection.

The Authentication (Auth) Module provides authentication of the correspondents on this application connection. The module has a common set of data structures as input to it. However, it is assumed that there will be many Auth Modules to choose from when assembling an Application. Authenticaion modules will define their own range of policies for a given authentication method. The Authentication module may generate its own protocol. Standard modules may be defined in the future.

5.3.3.3 Common Application Connection Establishment Procedure (CACEP)

The primary function of creating an application connection is to establish the set of objects to which remote operations on this flow have access. This is the fundamental definition of an AE. The Object-Set parameter in the Connect Request/Response allows the set of objects available on this connection to be restricted. The value of this parameter in the Response can be different than in the Request.

Connection establishment between members of a DAF occurs in the traditional manner. An Application-Process Instance issues an Allocate Request to its IPC Facility specifying the destination application process name, and the Quality of Service parameters it requires. If successful, the local IPC Facility returns a port-id to be used as a handle for all interactions with this flow.

Note that DAPs participating in a DAF may have multiple connections active with the same or different characteristics and with the same or different DAPs at the same time.

Once the IPC connection is established, the CACEP (CACE Protocol) request/responses are exchanged with the destination Application-Process. The Connect Request PDU will contain the initial information for the authentication exchange and the Connect Response PDU will contain the initial response[6]. If additional exchanges are required (as with some authentication techniques) these then follow. This allows for a range of authentication modules of various strengths. Once this procedure is complete, the correspondents have entered their data transfer phase and may commence with the function of the DAF.

5.3.3.4 Application data transfer phase

The data transfer phase of a DAF is entirely determined by the purpose of the DAF. There are four kinds of data transfer that can occur:

a) CDAP interactions with the RIB Daemon. This should be the vast majority of the traffic. Most tasks in a DAP will treat the RIB as local memory using the RIB to "page in" accessed information.

b) Direct CDAP exchanges between tasks in the DAF. This should be minimized since the use of a) is encouraged).

c) Encapsulating a legacy protocol, either with or without the use of the A-Data PDU.

d) Use of A-Unit-Data. Primarily used by DAFs that need to relay information among its DAPs.

Case a) represents the model that can be used for all distributed applications. The tasks of the distributed application are written as if accessing local data and the RIB Daemon makes the data available, much as memory management would in an operating system. Cases b) and c) correspond to the traditional view of networked applications sending requests and responses, transferring data, etc. This strategy has the advantage of isolating concurrency control to IPC and reducing considerably the possibility of deadlocks. This is the only viable future for sophisticated distributed applications.

Case d) would generally be used by telemetry flows in a DAF or DAFs that need to do relaying.

DAFs may relay, but not be able to impose an upper bound on Maximum Packet Lifetime. Hence, it is not IPC. Strictly speaking, the DAF is not limited to only interpreting or processing just the A-Unit-Data PCI, but may consider the entire PDUs in making a relaying decision, and there is no means to prevent DAPs from relaying PDUs.

5.3.4 DAF enrollment

For an application process to join a DAF, it shall be enrolled. Enrollment is carried out by the DAF Management task of the DAF infrastructure. Enrollment begins with an application process establishing an application connection with an existing member of the DAF. Once this management connection is created and the new member has been authenticated, the new DAP shall be initialized. This may include but is not limited to the following operations:

a) determining the current state of the new member (which may be a returning member);

b) determining the capabilities of or assigning capabilities to the new member;

c) assigning one or more synonyms to the new member for use with in the DAF;

d) initializing static aspects of the DAP, perhaps including downloading code; and initializing any DAF related policies;

e) creating additional connections to support distributed RIB operations;

f) initializing or synchronizing the RIBs;

Any other operations that would need to be performed before normal RIB Daemon updates can be relied on for maintaining the expected level of consistency and resiliency.

6 Introduction to distributed management systems (DMSs)

6.1 General

There are three forms of DMS:

a) A DAF Management system for remotely managing elements of one or more DIFs, is referred to in this document as a Network Management system or NM-distributed management system (DMS) (see 5.2).

b) A DAF Management system for managing a collection of DAFs used for a specific purpose, sometimes referred to as Application Management system or AM-DMS (see 5.3).

c) A DAF Management system for managing a given name space is a Name Space Management DAF, or NSM-DMS (see 5.4).

All processing systems, Application-Processes and IPC-Processes are managed by a DMS of some form. All resources used by non-DMS DAFs are monitored by these DMSs. The collection of elements managed by a DMS is called a DMS domain. There is no requirement that DAFs or DIFs be managed as a whole by a single DMS domain, although they may be. However, a DAP or IPC-Process shall be managed by one and only one DMS. There may be DIFs where every member IPC-Process falls under the domain of a different DMS. The only central or master role can be associated with initiating the DIF which itself was distributed. For example, initiating the creation of a DIF in response to a DIF Allocator request that required a new DIF over multiple (N-1)-DIFs would be managed by one DAP.

Most local DIF Allocator requests, i.e. those generated by DAPs on this processing system, will be forwarded to an NSM-DMS for actual resolution. The DMS responsible for the Applications (DAPs) or DAFs will maintain the local repository and will have the credentials to create IPC-Processes for joining existing or new DIFs. Hosts, on the other hand, would have greater use, but would probably maintain local caches of recently used applications to facilitate look up.

6.2 Network Management DMS (NM-DMS)

While the IPC-Processes that comprise the DIF are exchanging information on their operation and the conditions they observe, it is generally necessary to also have an outside window into the operation of DIFs comprising the network. While the members may reach a local optimization, it is often more complex to discover global optimizations without an “outside” perspective.

In these systems, control shall be automatic. Events are happening far too fast and state is changing too rapidly for a centralized system to be effective. Furthermore, the nature of distributed systems always opens the possibility for partitioning. Hence, it should be possible for distributed systems to fail-safe without central control. The purpose of Management is to monitor and repair, not to control.

A NM-DMS will perform the traditional functions of monitor and repair, e.g. deploying new configurations and monitoring performance. The DAF model can be applied to network management to represent the whole range from distributed (autonomic) to centralized (traditional). Both are required. As noted, experience with distributed algorithms has shown an ability to find local optima, but not always global optima. If nature guides, modest complexity begins to lead to concentrations of information processing, either in terms of ganglia or a true central nervous system. While today’s networks and distributed applications are closer to the less complex end of this spectrum, the architecture allows controlled investigation of the space. In particular, the fact that this architecture distributes the telemetry and management functions independently, makes it especially interesting and an excellent tool for investigating the trade-offs and what is possible.

In the traditional centralized network management architecture, an NM-DMS would be a heterogeneous DAF consisting of one or more DAPs providing management functions, with other DAPs providing telemetry. The management DAPs can be subdividing roles or tasks within network management or providing management for sub-domains and redundancy for each other. A typical DMS has the usual

tasks of event management, configuration management, fault management, resource management, etc. This also has the advantage of shifting the focus away from boxes to a distributed system model.

The NM-DMS DAPs has the traditional agent role function of collecting and pre-processing data. The Agent will have access to the DAF Management task of all IPC-Processes (and associated DAPs) in the processing systems that are in its domain. While there is no constraint, it is likely that an NM-DAF would have one “Agent DAP” for monitoring in each processing system with a DIF or DAF in its domain. The DAF Management task of each DAF or DIF in the NM-DMS domain is a kind of “sub-agent.” A Management Agent may be designed to seek out its DMS or alternate DMSs in the event of failures. There is no requirement that the NM-DMS shall operate over the topmost DIF in order to manage all of its assets, only that it operates over a DIF with sufficient scope to access all resources in its management domain.

As noted above, IPC-Processes are in one and only one management domain at a time. This means that only one NM-DAF has the ability to modify the behaviour of the IPC-Process.

It is possible for there to be multiple MAs responsible for different DIFs in the same processing system. For example, DIFs can be created as VPNs and can be allowedto be managed by their “owners”. The network designer shall be careful. There is only one processing system to meet the requirements of these MAs. One MA (and management system) shall be empowered to resolve conflicts or to bound the capabilities of other MAs.

In general, an IPC-Process in a DIF can be managed by one and only one manager at any particular time. Other managers may be given permission to read, i.e. observe, the system but not write to it, i.e. change configuration. Management systems will have mechanisms for defining management domains and changing their composition.

Network Management is characterized as “Monitor and Repair, but not Control”. The traditional monitor and repair functions are:

— Event Management (EM) – primarily handled by the RIB Daemon when unsolicited CDAP Writes arrive at the NMS. EM consists primarily of the authoritative log of all events and a subscription service that allows other management applications access to the information as it arrives.

— Configuration Management (CM) – supports the creation, maintenance, and activation of network configurations. Networks often have multiple configurations defined for a network, e.g. peak/off hour, holiday or weekend. Configurations will be often defined based on an existing configuration, rather than wholesale replacement of one configuration with another. In these cases, it is important to be aware of configuration-drift which naturally occurs the longer a configuration is in place. The starting point of a configuration transition may have moved in an incompatible way. This shall be resolved before activation begins. Activation is easily automated as a directed tree-walk.

— Performance Management (PM) - monitors the performance of the network by reading aggregated data from Management Agents and to a lesser degree the event stream. This information is generally not for immediate action. Immediate response is generally handled by the autonomic functions in the layers. Normally this data is used to further refine the policies being used by the network.

— Crisis Management (CrM) - primary use of the data coming in the event stream is for crisis management. The event stream data will be used primarily for detecting and countering denial of service attacks, as well as physical disaster associated with network assets, e.g. fire, flood. There can be alternate configurations for these situations that can be activated.

— Fault Management (FM) – one or more distinct Network Management Stations with very small management domains, i.e. the equipment being tested. FM also has specialized tools for generating traffic streams with given characteristics, probes for observing device behaviour, etc.

— Homonuculus – the set of applications used to support the operator control centre from which the central monitoring of the network is done. There is probably one of these per management domain and they should be able to back each other up.

— Coordination and Planning – the highest level of the network management planes that monitors at the long term and global health of the network. Analysis of performance and fault data is processed, and improvements proposed and tested before deployment.

The network is a low uncertainty enterprise and it is the job of network management to ensure that it stays that way.

6.3 Application Management DMS (AM-DMS)

Essentially all application management systems, such as they are, are point solutions. Examples include, e.g. utility management systems (gas, light and water), process control, factory automation. Process control should be one of the most untapped areas. The design and construction of large chemical or petroleum plants is fairly ad-hoc and the networking even more so. There is a major opportunity to introduce greater safety with architectures designed to contain accidents and prevent cascading catastrophic results.

The reason the DIF Allocation (and other functions) are not described as being unique to a processing system is precisely for AM-DMSs. It is easy to imagine the need to have an AM-DMS product such that it formed a complete package that managed its own IPC resources, even though these were provided by a NM-DMS managing its own DIFs.

6.4 Name Space Management DMS (NSM-DMS)

Management of a name space in a distributed environment requires coordination to ensure that the names remain unambiguous and can be resolved efficiently. Resolving a name as Saltzer noted in[5], requires "locating an object in a given context given its name". This requires a mapping from a name in name space, NS_A either to the object the name is bound to or to a name in NS_B, where B may be the same as A, such that resolving B (perhaps recursively) yields an object, i.e. indirection or aliasing. Hence, some systems (perhaps all) shall be given the authority to assign names.

For small, distributed environments, this management can be fairly decentralized and name resolution can be achieved by exhaustive search. Once found, the location of the information that resolved the name may be cached locally in order to shorten future searches. It is easy to see how, as the distributed environment grows, these caches would be further organized often using hints in the name itself, such as hierarchical assignment, to shorten search times. For larger environments, distributed databases may be organized with full or partial replication and naming conventions, i.e. topological structure, and search rules to shorten the search, requiring more management of the name space.

The NSM functionality can be in whole or in part, a function of any DAF. The functions of Name-Space Management are the following:

a) authenticate DAPs that are allowed request names to be resolved, i.e. users of NSM;

b) authorize and authenticate DAPs that are allowed to modify entries in the RIB about names (presumably only those for which it has management responsibility), including delegating authority for assignment of subsets of the name space;

c) assign an unbound name from the Name Space to an object;

d) assign an unbound set of names for future assignment by the requestor, i.e. delegation of naming authority, as well as its de-assignment;

e) implement policies for updating and maintaining the replicated RIB used by NSM, including the forwarding tables for facilitating requests;

f) resolve names by following the naming conventions and search rules;

g) check credentials of requests to determine if the name can be resolved; and

h) return a result that allows the NSM or the function created using it to complete its task.

This function may be a stand-alone DAF but more commonly will occur as a DAF within a DMS, a purpose specific DAF, or in a DIF. The two most common uses of the NSM functionality are as the DIF Allocator and the Flow-Allocator.

The primary purpose of an NSM is to resolve names in a timely manner. The result of this operation may be either an object, or another name. When the result is a name, this is what is normally called "indirection" and will require resolution of that result. The resulting name may be in the same name space or a different name space, which would mean consulting a different NSM. Strictly speaking, the NSM can only return a name. "Resolving to an object" shall be interpreted as resolving to a name that is unambiguous within the processing system where the object resides.

In smaller NSM-DAFs, the Server and Repository functions may be combined. The distinction between Servers that handle Query and Servers that handle Assignment may take several forms: minimal, differences in access control permissions, or distinct sub-DAFs. Because NSMs are often symbiotic DAFs, depending on the security requirements, implementations may integrate their functionality, e.g. RIB, into the host DAF.

Repository DAPs manage the distribution and replication of the RIB required to maintain efficient name resolution. Assignment-DAPs will, in most cases, be a function of another DMS. It is worth noting that in most cases the authority to manage aspects of a name space will not be given to arbitrary Applications but to DMS functions. There will undoubtedly be exceptions to this, however, modelling it this way will simply make those degenerate cases.

An Assignment DAP may also request to have a subset of the name space delegated to an NSM under its management. These can constitute entire child NSM of the name space. Clearly, in the case where a parent NSM delegates a naming authority for a subset of the namespace, e.g. a branch of the naming tree, this structure may be repeated in whole or in part as a child or sub-NSM-DAF. There is no requirement in the parent/child relation of NSMs that requires distribution and replication be limited to occurring within the child but may include parents and grandparents and other children.

The Name Space Management functionality is used to build specific namespace related services. It is incorporated into other Management DAFs or specific applications DAFs or DIFs as required. As described elsewhere, the Flow-Allocator is a use of NSM within a DIF. The NSM functionality can be used to create the Data Dictionary for a distributed or federated database.

6.5 DIF allocator

6.5.1 General

A major application of NSMs is the construction of a DIF-Allocator (DA). The DIF-Allocator arises naturally from the property, that at the IPC boundary, an Application merely requests that resources be allocated with the Destination by name. The Requesting Application does not need to know where the Requested Destination Application is. At the step in extending the IPC Model to the distributed case, when the case of N systems directly connected is considered, a function is required to maintain this property. While it is part of IPC, it is outside the DIFs to determine which "wire" (which DIF) to use. In Clause 4, this was IPC Resource management (IRM).

When an Application submits an Allocate Request to the IRM with an AP-Name, the IRM determines if the application is reachable on any of the DIFs available to this application and if so, one is selected and the Allocate is processed normally.

But if the desired application is not on one of the available DIFs, it is perfectly reasonable that this function queries its peer IRMs, which may have different DIFs with other applications on them. If it is found, then a new DIF can be created to enable the communication. This function is called the DIF-Allocator. It may span a set of DIFs and has two major functions:

a) to find the DIF that supports the requested DIFs or DAFs, that is not available to this IRM;

b) to make available a DIF by either joining existing DIFs or creating new DIFs such that both the requested and requesting Applications can allocate IPC. This can require the DIF Allocator to moderate (or negotiate) the creation of DIFs.

The DA communicates with its peer DAs and attempts to find the means to create a new “top level” DIF. Once it is created, the Allocate request is handled normally. For the DA to perform its function it shall use an instance of Name Space Management. Note that the name may be of a DAF or a DIF. In the latter case the IRM would attempt to join the named DIF. The DIF-Allocator is a ‘finder of new DIFs.’ A DIF-Allocator may be a stand-alone DAF, one embedded in a DMS, or within a DIF.

This is, in effect, the network generator function. While networks can be constructed by external ad hoc means, the DIF-Allocator provides the means for the recursive construction of networks organically based on user demand. Again, while it can appear to be, and it can be, quite useful, the RINA Model does not impose a requirement for a single authoritative DIF-Allocator, nor is there any means to impose one, nor does this model assume there is only one.

6.5.2 DIF allocator structure

If the IRM of every application were a member of a DIF-Allocator this can create a huge scaling problem. However, it is important to recognize that while every DAP or application process can need DIF-Allocator services at one time or another, there are other factors that impose constraints that mitigate this problem:

a) in a typical network, only a few IRMs will have a diversity of DIFs available to it, i.e. most IRMs will have to only one DIF or have access to the same DIFs;

b) not every application would have the permissions to register names;

c) not every application would have the permissions to allocate the resources for a new DIF.

These last two capabilities would more likely be assigned to Operating System DMSs (outside of the scope of this document), NM-DMSs, or Application-DMSs. By adopting this assumption, exceptions can be accommodated as degenerate cases. Thus, the vast majority of IRMs will only request DA services. This implies a structure for the DIF-Allocator DAF that not surprisingly mimics the structure of Name Space Management.

When the IPC Resource management (IRM) is requested to allocate IPC resources to a name, and it fails to find the requested application on any DIF it has access to, it will request a new DIF from DIF Allocator as a Client user (similar to the NSM Query-DAP). The DA will then manage finding a DIF that supports the application and the negotiations with the intervening DMSs with responsibility for the underlying resources to either allocate a new DIF or to join existing DIFs or both such that there is a single DIF that both the requesting and requested applications can use for IPC.

A DIF-Allocator is a DAF that incorporates an NSM and has a similar structure to an NSM. Since DIF-Allocator shall find applications that are not on adjacent DIFs. The DIF-Allocator DAPs (DA-DAPs) will need to relay requests and responses to where they can be processed. This implies that there are two sets of forwarding tables:

— a forwarding table, FT-s, for a name resolution request to forward to the next repositories, DA-r-DAPs, according to the search rules, and because the next NSM-r-DAP is not necessarily one hop away;

— a forwarding table, FT-n is required to forward PDUs through one or more of the DA-DAPs between the NSM-r-DAPs when the destination DA-DAPs aren’t one hop away.

Thus, there are two different graphs within the DA-DAF that are important: the graph defined by the search rules and the caching strategy of the repositories, and the graph of the connectivity of the DA-DAPs. The FT-s may organize the repositories into a flat, ring, hierarchical, or other structure. The graph implied by the FT-n may use a separate set of synonyms to facilitate forwarding PDUs from sender to destination, i.e. closer to traditional routing.

The components of the DA-DAF are much as they are in the NSM. The IRM of most applications will only be able to request a new DIF from the DIF-Allocator and have a DIF created for their use. In most cases, the DMS responsible for the domain in which Applications exist will perform all registration related functions. The registration process makes the names of applications and their supporting DIFs along with various properties available to the DA-DAF.

Some DAPs may be dedicated to maintaining the distributed database for resolving names. There is no requirement that these DAPs are directly connected. The FT-s forwarding tables define the connectivity of the search, but the FT-n may be required to forward PDUs through intermediaries to get to the next place to search.

The various functions are not all required in all DAPs and some of these may be used for isolating external functions from internal operation. The DAPs supporting Query and Registration Server functions are shown as complete to indicate that local caching may be present. Some environments may choose to combine functions and still meet their requirement.

While this description has referred to applications, the DIF-Allocator may be used to find DIFs as well as DAFs. Also, while this description implies that DIFs are always created, they may not be. In some cases, the DA-DAF may be able to create the necessary "top layer" by instructing IPC-Processes to join an existing DIF.

6.5.3 Operations

When the DA is requested to create a DIF for a pair of applications, it first has to find the desired DAF. The flow allocation request will be augmented with parameters that a DIF shall provide. These will either be provided by the appropriate DMS or derived from the parameters of the available DIFs as well as the policies the DMS has on DIF creation. The IRM requests a Create DIF specifying the application name that it shall support and the ranges (tolerance) of QoS parameters, not the QoS-cubes. This begins as a traditional search using the common Name Space Management functions. Although this can require relaying the request through a number of DAPs to get to the next Repository, once a DIF is found that supports the application, the DA shall negotiate with its DMS and the DMSs of any DIFs required to support a new DIF or direct systems to join an existing DIF.

Unlike functions such as DNS or X.500, a DA-DAF search is concluded only when the management domain responsible for the requested application or DIF is found, i.e. the search terminates when the credentials of the requesting Application can be checked to determine if the requestor has access to the supporting DIF. This also ensures that the security of the application is not compromised, by returning whether or not the application exists. If the requesting application's credentials do not allow access, "not found" is returned. Once a successful search is concluded, the next step is to create a DIF that supports at least the requesting and requested applications. This may be done either by joining existing DIFs or by creating a new DIF or by some combination of creating and joining DIFs.

Given that creating IPC-Processes to join existing DIFs or to be members of new DIFs requires the allocation of significant resources and the permissions to do so, the DA-DAF will act as the mediator in conjunction with the AM-, NM- and Operating System-DAFs responsible for appropriate resources to extend existing DIFs or to create new ones. The DA-DAF will request the appropriate Management DAF to create of IPC-Processes in specific processing systems and instruct them to join a DIF. Clearly, the Create-DIF request will have to include parameters for the new DIF. This may involve parameters such as (but not limited to): QoS-cubes supported, minimum average degree of the DIF graph, cost. When the new DIF is created, the DIF-Allocator will return a response to the requesting IRM, which may proceed with its Allocate Request. The DIF-Allocator may also spontaneously notify the IRM of a new DIF.

Creating a DIF within a single Management Domain simply skips the considerations of negotiating with different management domains and goes straight to creating the IPC-Processes within the Management Domain for the new DIF.

Algorithms for creating DIFs with various properties will be a topic of research for some time to come. Of special interest will be algorithms for creating DIFs with average degrees greater than two. However, the straightforward cases can be considered. In creating a new DIF, a new IPC-Process will need to be

created at Border Routers of the existing top rank DIFs. The following are obvious approaches to be taken:

— The DIF-Allocator having found a DIF, B, with the requested Application, creates an IPC-Process on the processing system that supports the application using the DIF B as the (N-1)-DIF. The DA then sends a response back, creating an IPC-Processes at the Border Routers along the return path of the Create DIF Response.

— The DIF-Allocator DAP, which found the DIF with the requested application, consults its RIB and constructs a set of nodes to be in the new DIF. It creates a new IPC-Process to serve as the seed for the new DIF. It then sends Create IPC-Process commands to each node in the set with instructions to join the New DIF.

There is also the null case or bootstrap case, where a processing system is joining its first DIF. There are three cases to be considered:

a) The use of a Shim DIF: For existing legacy media protocols, RINA employs a Shim DIF. A Shim DIF provides the minimal functionality necessary to make an existing media standard have the same API behaviour as a DIF with the properties of this media. A Shim DIF makes no attempt to "enhance" the existing media protocol. In this case, enrolment will follow the procedures of the legacy protocol. Normal DIF operations will work above that.

b) A DIF operating directly on point-to-point media: In this case, it is assumed that there is either some ad hoc first PDU or that the process on the end of the point-to-point media, i.e. a wire, is expecting a RINA enrolment procedure. This will begin with CACE-Connect, also referred to as an M-Connect. Given these conditions the procedure can progress with normal enrolment.

c) A DIF operating directly over a multi-access media: In this case, it is assumed that some unique identifier, e.g. an equipment serial number, is available to distinguish correspondents at the other "end" of the media. Either an ad hoc first PDU will carry this information or the "other ends" will be expecting a CACE-Connect PDU with this "unique identifier" as the Destination Application name. The Source Application name will also have to be known to be unique within the scope of the media. The source unique identifier shall identify the IPC-Process that is requesting to join the DIF. The destination unique identifier may name either the DIF being joined or an IPC-Process that is a member of the DIF.

This exchange can now be used to either create a DIF on top of the media or join an existing DIF using the normal procedures, including the assignment of addresses within this DIF. However, the "unique identifiers" will still be required to distinguish traffic between different DIFs on the same media and within the scope of the media. There is, in effect, a very minimal Shim DIF over the multi-access media itself.

7 Distributed InterProcess Communication — Fundamental structure

7.1 Distributed-IPC-Facility (DIF)

7.1.1 Overview

A Distributed-IPC-Facility, the primary focus of this model, is a homogeneous DAF that provides and manages IPC. To avoid confusion, the members of a DIF are called IPC-Processes. They are no different than DAPs (the members of a DAF) or any other application processes, other than that they are members of a DIF.

A Distributed-IPC-Facility is a distributed application that manages IPC and consists of at least one IPC-Process in each processing system participating in the DIF. A DIF is a layer. The scope of a layer is the set of cooperating IPC-Processes (that maintain distributed shared state) that comprise a DIF. In this document, DIF and layer are used interchangeably. For IPC and networking, there is one type of layer that repeats as many times as necessary to effectively cover the range required for the operation of the

network. Layers in this architecture are divisions of ranges of allocation and scope. This leads to the following formulation of the properties of layers:

— Different layers tend to have a different scope.

— There may be multiple layers of the same rank, either in parallel or serially.

— Functions do not repeat with the same layer at the same scope.

— Groupings of functions within a layer tend to be by an orthogonal criterion.

— There should be strong interfaces at layer boundaries that abstract the internal functioning of the layer.

— If there are layers of different rank with the same scope, their functions shall be independent.

Each DIF only manages its own resources. (N+1)-flows are multiplexed into (N)-flows. They are aggregates of traffic of the same or different (N+1)-QoS-cubes mapped onto (N)-QoS-cubes. Regardless, the QoS-cubes of the (N)-DIF are different than the QoS-cubes of (N+1)-DIF and the resource allocation problems the DIF shall solve are different. To speak of (N)-flows in DIFs at ranks less than N is meaningless and counter-productive.

Because there can be more than one DIF of the same rank, there is no direct IPC between different (N)-DIFs, i.e. DIFs of the same rank, without relaying above. IPC between DIFs of the same rank within the same processing system shall use either an application process with only local knowledge (sometimes called a protocol converter), or by an application process with knowledge of a wider scope, e.g. relaying by an IPC-Process of a (N+1)-DIF.

Watson[9] worked out the necessary and sufficient conditions for synchronization for reliable data transfer (IPC). Those conditions were surprisingly enough to bound three timers:

a) Maximum Packet Lifetime (MPL);

b) Maximum Time to Wait before Acknowledgement (A);

c) Maximum Time to Exhaust Retries (R).

While the definitions in the model are written to be general, this model indicates that a single Error and Flow Control Protocol that separates mechanism and policy and is defined independently of a specific encoding, is not only possible but beneficial. As with the DAF, where a single application protocol was found, there is a single data transfer protocol for the IPC.

7.1.2 IPC process

A DIF is a Distributed-Application Facility (DAF) that is designed for a specific purpose, i.e. providing IPC services over a given range of operation. Hence, an IPC-Process has all of the elements of a DAP as described in Clause 6. The IPC-Process structure naturally cleaves into three distinct loci of processing with decreasing duty cycles as one moves left to right decoupled through a repository for state information.

The **DAP Infrastructure** consists of:

— DAF Management – the local task concerned with the overall management of the DAF accessed by the DAF Management Agent for this Processing-System.

— Scheduling (e.g. Resource Allocation, routing).

— Memory Management – the RIB Daemon that stores routing tables and other local state information (the MIB).

— IPC Management – responsible for the use of the supporting DIFs, including SDU Protection, multiplexing, and the DIF-Allocator.

The Tasks of the DAP, in this case the functions necessary to provide IPC, are:

a) SDU Protection, which does data corruption detection, data confidentiality and integrity, time to live, etc. There is potentially a different SDU-Protection for each (N-1)-port.

b) Potentially, a different Multiplexing Task (MT) for each (N-1)-port that maps (N)-PDUs to the appropriate (N-1)-port.

c) A single Relaying Task, which is concerned with managing the utilization of the layer below.

d) Flow-Allocator Task, which allocates and manages each flow when requested via the API.

e) An Error and Control Protocol Machine for each IPC flow supported, consisting of

   1) a Data Transfer task for each IPC flow being supported, which distinguishes flows and performs sequencing, fragmentation/reassembly, etc, as necessary;

   2) a Data transfer control task for each IPC flow that requires retransmission and/or flow control mechanisms,coordinated through a state vector.

f) Delimiting, which ensures that the identity of the SDU can be maintained and maps the SDU to the user-data field of the (N)-PDU.

g) The Application Programming Interface (API) to request, use and release IPC resources.

All instances within an IPC-Process have the same concrete syntax and range of policies.

7.1.3 Flows and connections

One of the major implications of Watson's work is that port allocation and synchronization should be decoupled. In RINA, flows and connections are decoupled. The Flow-Allocator allocates ports and creates the flow. Port-ids are FAI-ids and are the endpoints of a flow. When there has not been data on a flow for 2Δt, the shared state exists between the EFCP-instances. This is called a connection. Connection-endpoints (CEPs) exist in the EFCP-instances. The connection is between communicating EFCPM-instances. A flow consists of the bindings between the source and destination port-ids and the source and destination connection-endpoint-ids respectively, and the connection or the potential for a connection to exist.

The Flow-Allocator is responsible for allocating the port-ids and creating the EFCP-PM-instances. Port-ids are only deallocated when a Deallocate API call is made.

7.1.4 DIF security principles

The primary task of (N)-DIF security is to protect itself from malicious attacks or the collection of sensitive information on its operation. A (N)-DIF can assume that the (N+1)- and (N-1)-DIFs are doing the same.

There are five security services that may be supported by a DAF: Authentication, Access Control, Confidentiality, Integrity, and Non-Repudiation. The last applies only to DAFs. The others apply to all DIFs, as necessary.

— All members of a (N)-DIF should be authenticated as legitimate members of the DIF. This is part of the Enrollment Phase.

— While Access Control most commonly appears in the layer directly supporting Applications, it may occur in lower layers.

— Confidentiality and Integrity counters corruption, replay, and eavesdropping of the contents of (N)-PDUs and should be used when the (N)-DIF has generated sensitive information that may be of use to an intruder keeping in mind that multiple encryptions can weaken the strength of the result.

The above implies that DAFs should protect themselves. This implies that the only threat to confidentiality and integrity to a DIF directly supporting DAFs can be Traffic Analysis and below that and toward the core of the network, there is little need for confidentiality.

Given that many lower ranking DIFs have limited scope and can operate within a controlled environment, the degree of security should be capable of being selected, from none to very strong.

When a (N)-PDU is passed as an (N-1)-SDU, and if confidentiality measures are in place, the (N-1)-SDU cannot be interpreted by the (N-1)-IPC_Process, i.e. no elements of the (N)-PCI are clear-text.

The length of a (N)-address should be a small multiple of the maximum number of members of the (N)-DIF. Avoid overloading the semantics. If at all possible, the (N)-address should only be assigned when the (N)-IPC_Process joins the (N)-DIF.

Any (N)-identifier shared between an (N+1)- or (N-1)-DIF should not be carried in protocol. Such identifiers should be local to the (N)- and (N-1)-IPC_Process.

No identifier should be used for more than one purpose. It should have a single semantics.

An (N)-address should not be visible to the (N+1)- or (N-1)-DIFs. A complete (N-1)-address should not be used as a component an (N)-address.

Multi-level security is an DAF issue. Any method to support multiple levels in the DIFs below violates security by distinguishing multiple levels of security.

7.1.5 IPC components

7.1.5.1 General

The IPC-Process has two major functions:

— supporting IPC requested by applications by creating one or more (N)-connections for each Allocate-request;

— managing its use of one or more (N-1)-connections with one or more DIFs.

7.1.5.2 IPC API

The IPC Service Primitives are used by an Application-Process to request IPC-Facilities. The most common form of the primitives is:

— Reason <- Allocate_Request (Destination, Source, QoS Parameters, Port-id) – an asymmetrical request/response issued by an application to the IPC-Facility to create an instance of a connection with an application, or by an IPC-Process to an Application-Process to notify it of a request for communication.

— Reason <- Allocate_Response (Destination, QoS Parameters, Port-id) – an asymmetrical request/response issued by an application to the IPC-Facility to respond to a request to create an instance of a connection with an application, or by an IPC-Process to notify the requesting application process of the result of its request.

— Reason <- Send (Port-id, buffer) – issued by either application process to send an SDU to the destination application on this port.

— Reason <- Receive (Port-id, buffer) – issued by either application process to receive an SDU from the destination application on this port.

— Reason <- Deallocate (Port-id) – issued by either Application process or DIF to request or notify of the deallocation of the resources held for this allocation and destroys all the shared state associated with it and notifies the communicating applications.

Allocation is the function that occurs at the API (layer) boundary with the DIF. Applications request the allocation of communication resources. The DIF then determines how to provide those resources, which will require assigning port-ids, allocating resources, and instantiating instances of the EFCP.

7.1.5.3 Flow-Allocator

The Flow-Allocator is responsible for creating and managing an instance of IPC, i.e. a flow[8]. Logically, there is an instance of a Flow-Allocator for each allocated flow. A component of Name Space Management for this DIF exists in this IPCP to manage the assignment of addresses and to resolve the mapping of (N+1)-names/addresses to (N)-names/addresses of an IPC-Process. This function is also accessed during enrolment of a new member of the DIF to obtain an address for use internal to the DIF. The policies of the NSM may also delegate blocks of addresses to members of the DIF.

The IPC-API communicates requests from the DAP to the DIF. An Allocate_Request causes an instance of the Flow-Allocator to be created. The Flow-Allocator-Instance (FAI) determines what policies will be utilized to provide the characteristics requested in the Allocate. The FAI instantiates the EFCP instance for the requested flow before sending the CDAP Create Flow Request to find the destination application and determine whether the requestor has access to it.

A create request is sent with the source and destination application names, quality of service information, and policy choices, as well as the necessary access control information. Using the Name Space Management component of the IPCP, the FAI shall find the IPC-Process in this DIF that resides on the processing system that has access to the requested application.

This exchange accomplishes three functions:

a) following the search rules, the Name Space Management function finds the address of an IPC-Process with access to the destination application;

b) determining whether the requesting application process has access to the requested application process and whether or not the destination IPC-Process can support the requested communication;

c) instantiating the requested application process, if necessary, and allocating a FAI and port-idin the destination IPC-Process.

The create response will return an indication of success or failure. If successful, destination address and connection-id information will also be returned along with suggested policy choices. This gives the IPC-Processes sufficient information to then bind the port-ids to an EFCPM-instance, i.e. a connection, so that data transfer can proceed.

In addition to creating flows, when an IPCP of the DIF changes address and this IPCP is the source or destination of a flow with that IPCP, the Flow-Allocator manages the change of address[8]. Also, the Flow-Allocator manages the replacement of connections for the same flow to avoid repeating the same sequence numbers with the same connection-id and thus maintain the security of the flow.

7.1.5.4 Delimiting

A DAP will deliver an amount of data to a DIF, called an SDU or service-data-unit. An SDU is a contiguous piece of data. Unless otherwise directed, the DIF will deliver the same SDU to the recipient. To do this, the SDU may be either fragmented or concatenated with other SDUs to fill the user-data fields of the PDUs. Hence, the first operation on an SDU will be to delimit it so that its identity is preserved and create the user-data fields for the PDUs to be sent. This operation is specifically designed not to be part of the EFCP Protocol. There is one Delimiting function per (N)-flow.

7.1.5.5 Error and Flow Control Protocol (EFCP)

7.1.5.5.1 General

The EFCP is strongly based on Watson's delta-t[12]. This protocol provides the IPC connection/flow associated with each allocation request. An EFCP connection operates strictly between the EFCPMs. The EFCP connections are then bound to specific AEs in the source and destination APs. This binding is made after a successful response to FAI request to create a flow. There is one EFCP instance for each (N)-Flow.

7.1.5.5.2 IPC Data Transfer Procedure (DTP)

The EFCP-PM operates on a single data transfer PDU with the tightly-bound IPC mechanisms. The PDU contains only source and destination addresses, QoS-cube-id, and connection-endpoint-ids, the latter used as the connection-identifier, a PDU-id (message-id or sequence number), and an optional checksum. For some IPC connections, this is the only PDU required, i.e. traditionally referred to as Unit-Data. Interactions between flows are done by the RMT and IPC Management Task. The only change to this protocol to support more robust connections is that the PDU-id is interpreted as a sequence number.

7.1.5.5.3 IPC Data Transfer Control Procedure (DTCP)

The DTCP provides the loosely-bound mechanisms as required. There is a distinct instantiation of this procedure for each Allocation request that requires retransmission or flow control. Flow and retransmission control can be utilized alone or together. Coordination of DTP and the DTCP is done through a shared state vector. DTP writes the state vector, while DTCP reads the state vector and generates control PDUs and controls flows at the DIF API. Interactions between flows are done by the RMT and resource allocator of the IPC Management Task.

7.1.5.6 Relaying Task (RT)

An IPCP of a DIF will contain zero or one Relaying Task (RT). If an IPCP in an (N)-DIF contains an RT, there will be no RTs in IPCPs of lower rank in the same processing system and no RTs in IPCPs in the same processing system of higher rank.

RTs come in three forms depending on where they occur in the architecture:

a) an RT is generally not found in hosts and the lower layers of routers;

b) an RT primarily found in the "top" layer of interior routers;

c) an aggregation RT primarily found in the "top" layer of border routers.

The Interior Router RT handles transit flows. It shall have high performance and minimal processing overhead. This process most closely resembles traditional packet forwarding. It receives PDUs from an input queue, inspects the destination address and QoS-cube-id, refers to a forwarding table and sends PDUs as quickly as possible to an output queue. Lower layers of an interior router will have RTs similar to a host. PDUs have already been concatenated and assigned to flows.

The Border Router RT does relaying, but also manages high traffic intermediate flows. Traffic with similar or complementary QoS requirements and common intermediate paths are aggregated onto these flows and the (N+1)-PDUs may be aggregated to increase efficiency and lower switching overhead. The primary difference between an interior and border router is that a border router does an extra level of multiplexing and may aggregate PDUs.

7.1.5.7 Multiplexing Task (MT)

The primary purpose of this task is to moderate the multiplexing of PDUs of different (N)-connections onto an (N-1)-port. Strictly speaking, the multiplexing function is inherited from the general DAP

structure and there is potentially one multiplexing function per (N-1)-port. The MT is responsible for the delivery of PDUs to the lower layer. The management of the lower layer interface is the responsibility of the IPC Resource management infrastructure task. The MT does not generate any additional PCI.

7.1.5.8 SDU Protection

SDU Protection has the same functionality as described as part of IPC Management in the common DAP infrastructure. The only slight difference is that an IPC-Process may have more than one supporting DIF. The SDU Protection on each (N-1)-flow may be different. This will primarily occur where the (N-1)-DIFs shall reflect the limited characteristics of the media.

7.1.6 DAF Infrastructure for IPC

7.1.6.1 DAF Management for IPC

The DAF Management Task of an IPC-Process has the corresponding responsibilities found in any DAF:

— Enrollment – To join a DIF (or any DAF) a potential new member shall first enrol. Using the name of the DIF, a potential new member IPCP establishes a CDAP connection using an (N-1)-DIF with another IPC-Process, which is already a member of an existing DIF. The establishment procedure with the member IPC-Process includes authenticating the newcomer, initializing various managed objects and their attributes including assigning the newcomer an address. Similarly, the new IPC-Process will have new information to share with the DIF.

— Security Management – A DIF requires three security functions:

  — authentication to ensure that an IPC-Process that wants to join the DIF is who it says it is and is an allowable member of the DIF (Enrollment);

  — confidentiality and integrity of all SDUs (SDU Protection);

  — access control to determine whether application processes requesting an IPC flow with a remote application has the necessary permissions to establish communication. (Resource Allocation).

— DAF Management: performs authentication of new members, as well as key management and other security management functions required for the security measures. Access Control is performed by the Flow-Allocator. The particular security procedures used for these security functions are a matter of policy. SDU Protection provides confidentiality and integrity.

7.1.6.2 Scheduling and resource allocation

The Resource Allocator is responsible for the following functions:

a) manage enrolment;

b) monitor the use of resources within the DIF by exchanging information with other IPC-Processes;

c) manage the assignment of new flows to the RMT;

d) regulate the resources allocated to individual flows;

e) manage the creation of the Forwarding Table;

f) monitor the RMT;

g) manage the service with all (N-1)-DIFs.

Virtually all of this task will need to be policy. The most that can be done is to provide a rich (or the right) collection of variables required for good decisions.

7.1.6.2.1 Resource Allocator

If a DIF is to support different qualities of service, then different flows have to be allocated with different policies and traffic for them to be treated differently. To meet the QoS requirements, different resources have to be allocated to different flows and information about the allocations distributed to the members of the DIF. There are three classes of such flows:

a) flows requested by an Application-Process, using this DIF;

b) flows with (N-1)-DIFs created by IPC Management (the IRM) for distinct classes of QoS to aggregate traffic and enhance efficiency, generally in border routers and among them;

c) flows that transit a system, i.e. traditional routing.

The primary task of the Resource Allocator is to monitor and coordinate the IPC Tasks, IPC Management, and the Flow-Allocator to ensure that quality of service targets is being met and to adjust policies according to changing conditions. This may be directed by a Network Management system responsible for the IPC-Process.

7.1.6.2.2 Routing

A major input to the Resource Allocator is Routing. Routing performs the analysis of the information maintained by the RIB to provide connectivity input to the creation of a forwarding function. To support flows with different QoS in current terminology requires using different metrics to optimize the routing. However, this shall be done while balancing the conflicting requirements for resources. Current approaches can be used but new approaches to routing are required to take full advantage of this environment. The choice of routing algorithms in a particular DIF is a matter of policy.

7.1.6.3 RIB Daemon

The RIB Daemon is the logical repository for information relating to the state of the DIF and the IPC-Process in particular. As such it can be viewed logically as a partially replicated distributed database. All state information maintained by the IPC Tasks, the Flow-Allocator, Resource Allocator, etc. is maintained by and available through the RIB Daemon. This includes all local information on the operational state of the DIF, performance, load, routing update, directory caches, etc. Besides making this information available to the tasks of the IPC Process, it is also the task of the RIB Daemon to efficiently update this information with other members of the DIF and the Network Management system periodically or on certain important events. There is no requirement that the same update strategy be used for all information in the DIF. The RIB Daemon is the primary user of CDAP within the DIF as it is in a DAF. The Network Management system also has access to the RIBs of all IPC-Processes in all DIFs in its Domain.

7.1.6.4 IPC Management

As with any other DAF, an IPC-Process has an IPC Resource management (IRM) task to manage its use of its supporting IPC. IPC Management should monitor its (N-1)-Flows. Notifying the Resource Allocator of major changes and where appropriate to find new (N-1)-DIFs. Strictly speaking, SDU Protection and Multiplexing are part of this task. However, observing this distinction too rigorously can lead to poor implementations.

Each DIF is concerned with three name management problems:

a) the mapping of (N+1)-names to (N)-Names, i.e. addresses;

b) the connectivity among (N)-addresses, i.e. next hops;

c) the mappings of (N)-addresses to (N-1)-names or port-ids, i.e. points of attachment.

a) and b) will utilize the appropriate NSM functions, while c) is generally referred to as routing, although forwarding is more accurate.

The NSM function in a DIF is closely related to and part of creating the forwarding table used for routing. For those cases where the only applications that are accessible via a given (N-1)-DIF are the IPC-Processes of a (N)-DIF, the NSM information is the information required to choose different paths to a next hop. Hence, this function can be viewed as part of routing.

7.2 Naming in Distributed IPC Facilities

An IPC-Process has at least four basic kinds of AEs:

— Flow-Allocators that manage the creation and maintenance of flows;

— EFCP-instances, that create IPC channels to APs which are users of this DIF;

— Data Transfer consisting of the RMT and SDU Protection;

— Resource Allocator to manage the flows associated with the internal management of the DIF.

The following identifiers for a DIF are needed:

a) A FAI is created when a request for allocation is received from a using Process. The FAI-identifier is returned to the requestor as a handle for all operations relevant to this request (e.g. Status, Read, Write) Traditionally this is referred to as a port-id. The port-id shall be unambiguous within the scope of the IPC-Process.

b) An EFCP-instance identifier unique with within the scope of the EFCP. There is one for each flow created to support the service request in a). Traditionally, this identifier is called a connection-endpoint-id and identifies one end of the shared state with its apposite.

c) The Data Transfer AE consisting of the RMT and SDU Protection is named purely for management purposes.

d) The Resource Allocator manages the sharing of information with other members of the DIF. In terms of the model, there may be one instance for each (N-1) DIF used by this IPC-Process.

In addition, there are three additional identifiers that are useful:

e) Because the AP-name is global in scope, its use for coordination purposes within the DIF would be inefficient both because of the length and that the structure of the AP-name is created to be effective at finding the object in the global space of all APs. Therefore, it is very useful to create a name space of synonyms for the IPC-Processes that are unambiguous only within the scope of the DIF. These will be much more efficient to use both because they are shorter and because they can be structured for use within the DIF. Traditionally, when structured to facilitate routing, this has been called an address. Other synonyms may be applied for other purposes.

f) Since there can be multiple allocations between the same pair of IPC-Processes, it is necessary to distinguish them. A unique identifier is created by concatenating the data transfer AE-instance-identifiers of the source and destination IPC-Processes described in b) above. This has been traditionally called a connection-identifier, and the data transfer AE-instance-id, a connection-endpoint-identifier. It is unique with the scope of the source and destination data transfer AEs.

g) A Distributed-Application-Name to designate not only DIFs, but other distributed applications. This is a whatevercast name for the set of members of the Distributed-Application. Its scope is the name as the scope of the AP name space.

Other than names of sets of AP-names and addresses as described in Clause 5 that may be used for various purposes, names for no other objects are required for a functioning DIF or for a network (see Table 1). All identifiers except addresses have scope local to the IPC-Process.