INTERNATIONAL STANDARD

ISO/IEC 4396-7

First edition
2023-12

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

Part 7: Flow allocator

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

Partie 7: Allocateur de débit

Reference number
ISO/IEC 4396-7:2023(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

This document defines the sequencing of the interactions of the flow allocator (FA). It is not, strictly speaking, a protocol specification. The protocol used in this document is the common distributed application protocol (CDAP). This document uses the objects required to create a flow between two processes and bind their endpoints to the applications that requested the flow.

The flow allocator is responsible for creating and managing an instance of interprocess communication (IPC), i.e. a flow. The IPC-API communicates requests from the application to the distributed IPC facility (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. It is important that how these characteristics are communicated by the application is decoupled from the selection of policies. This gives the DIF important flexibility in using different policies, but also allows new policies to be incorporated. The FAI creates the error and flow control protocol (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 (NSM) function, the FAI searches the IPC process (IPCP) in the DIF that resides on the processing system that has access to the requested application. This exchange accomplishes three functions:

— follows the search rules using the NSM function to find the address of an IPC-Process with access to the destination application;

— determines 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;

— instantiates the requested application process, if necessary, and allocate a FAI and port-id in the destination IPCP.

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 EFCP-instance, i.e. a connection, so that data transfer may proceed.

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

Part 7: Flow allocator

1 Scope

This document provides the flow allocator (FA) specification. It includes an overview of the flow allocator, its service definition, and its specification.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 4396-1, *Telecommunications and information exchange between systems – Recursive Inter-Network Architecture – Part 1: Reference Model*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 4396-1 and the following 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
directory forwarding table
table that maintains a set of entries that map application naming information to interprocess communication (IPC) process addresses which belong to the IPC Processes, where the requested application can be found or has information about where to search next

3.2
error and flow control protocol instance
EFCP instance
instantiation of EFCP for managing a single connection

3.3
Quality of Service-cube-id
QoS-cube-id
identifier unambiguous within this distributed IPC facility (DIF) that identifies a QoS-hypercube

Note 1 to entry: As QoS-cubes are created they are sequentially enumerated. QoS-cube-id is an element of Data Transfer PCI that may be used to classify protocol-data-units (PDUs).

4 Overview of the flow allocator

4.1 Narrative description of the service

This task creates and monitors a flow and provides any management over its lifetime. Its only service is to network management.

4.2 Narrative description of the task

The flow allocator has three major functions:

a) the allocation of a flow, i.e. an instance of communication;

b) to change the source address as instructed by the Resource Allocator;

c) to change the connection-id when the SequenceNumberRolloverThreshold is reached to next value in those provided during Allocation.

5 Service definition

When an Allocate_Request or Deallocate_Request API call is made, it is processed by the flow allocator. See 6.2.1.

6 Detailed specification of the task

6.1 Common elements

6.1.1 Directory forwarding table

The directory forwarding table maintains a set of entries that map application naming information to IPC process addresses. The returned IPC process address is the address where the requested application can be found or has information about where to search next.

If the returned address is the address of this IPC Process, then the requested application is here; otherwise, the search continues, i.e. either this is the IPC process through which the application process is reachable, or it can be the next IPC process in the chain to forward the request. The Directory Forwarding Table should always return at least a default IPC process address or Application Process Naming Information to continue looking for the application process, even if there are no entries for a particular application process naming information known to this DIF. An IPC Process may maintain alternate tables to be used under special circumstances.

The form of the table is:

Destination Application Process Name, Address of next place to look.

6.1.2 Flow object

The information necessary to create a flow object and send in a CDAP *M_CREATE* Request PDU is shown in Table 1.

Table 1 — Attributes in flow object, summary

Attribute name	Type	Description
Source_Naming_Info: *Application-Process-Name*: String *API-identifier*: String Optional *AE-identifier*: String Optional *AEI-id*: String Optional	Struct	The naming information of the application that requested the flow.
Destination_Naming_Info: *Application-Process-Name*: String *API-identifier*: Integer Optional *AE-identifier*: String Optional *AEI-id*: String Optional	Struct	The naming information of the application that is the destination of the flow.
Source-Port-Id	Unsigned integer	The handle of the flow service for the source application.
Destination-Port-ID	Unsigned integer	The handle of the flow service for the destination application.
Source-Address	Unsigned integer	The synonym of the IPC Process that provides the flow service to the source application.
Destination-Address	Unsigned integer	The synonym of the IPC Process that provides the flow service to the destination application.
Connection-ids: *QoS-cube-id*: Unsigned integer *Destination-CEP-id*: Unsigned integer *Source-CEP-id*: Unsigned integer	Struct array	This is an array of Connection-ids identifying the connections that may be active for this flow, generally no more than two. Connections that are going to be used earlier have a lower index, i.e. the first connection-Id to be used is the one in the 0 position.
Current-Connection-id	Unsigned integer	This is the index of the Connection-id that is currently "active" for sending data, although data may still arrive on earlier connections.
State	Byte	The state of the flow.
QoS-Params: *AverageDataRate:* Unsigned integer *AverageSDUDataRate*: Unsigned integer *PeakDataRateDuration*: Unsigned integer *PeakSDUDataRateDuration*: Unsigned integer *UndetectedBitErrorRate*: Double *PartialDelivery*: Boolean *Order*: Boolean *MaxAllowableGapSDU*: Unsigned integer *Delay*: Unsigned integer *Jitter*: Unsigned integer *extraParameters*: Array of name-value pairs	Struct	The list of parameters from the *Allocate_Request.submit* call that generated this flow.
Policies	Array of name-value pairs	The list of policies that are used to control this flow. This is conceptually a property list of (name, value) pairs, where the name represents a policy, and the value represents the name of a policy choice.

Table 1 *(continued)*

Attribute name	Type	Description
Policy-Parameters	Array of name-value pairs	This is a list of values that are arguments to the selected policies, used to set operational parameters for this flow. Conceptually, these are name-value pairs, where each name specifies the name of a parameter for one of the selected policies, and the value provides its initial value.
Access Control	Capability	This can be used to represent the local owner of the flow and the rights he grants to others to affect the flow, so that local System/Network Management can apply a discretionary access policy when responding to, for example, a request by a user to terminate some flow.
MaxCreateFlowRetries	Unsigned integer	Maximum number of retries to create the flow before giving up.
CreateFlowRetries	Unsigned integer	Current number of retries.

6.2 Specification

6.2.1 General

The distinction between a FlowAllocator and a FlowAllocator-Instance is significant. This document deviates slightly from the pure definition. The FlowAllocator is considered to be the container or manager for all FAIs. Hence, it has functionality that is independent of and distinct from the functionality of an FAI. The functionality of an FAI is concerned with the creation and management of a single allocation over its entire lifecycle. The FlowAllocator is responsible for functions involving allocates in general, i.e. functions that affect all FAIs.

The resources available to the flow allocator and its allocation policy are managed by the Resource Allocator. In some sense, all Allocate and Deallocate API calls are first handled by the FlowAllocator. The flow allocator instantiates a flow allocator Instance to manage each flow. Note that while this document only refers to create and delete operations, system management or other functions may perform operations on a FlowAllocator and its instances as well.

NOTE The action of the FAI can involve more than one EFCP connection. Also, flows are not necessarily point-to-point. An Allocate_Request that has application naming information that refers to a set, i.e. whatevercast, would result in the allocation of a multipoint flow. Other than the parameter referring to a set there is no difference to the user of the API.

6.2.2 Behaviour of a flow allocator

6.2.2.1 Initialization

When the flow allocator is initiated, it shall subscribe to the following CDAP Requests with the RIB Daemon and any other events relating to the management of individual flows:

Create_Request Flow

Create_Reponse Flow

Delete_Request Flow

Delete_Response Flow

Write Current Address

SequenceNumberRollOverThreshold Events

Update DirectoryForwarding Table, period X seconds, on Events MembersLeftDIF

6.2.2.2 Allocate_Request.Submit

6.2.2.2.1 When invoked

This service primitive is invoked by an Application Process not a member of this DIF to request the allocation of IPC resources with a given destination Application Process.

6.2.2.2.2 Action upon receipt

The flow allocator is invoked when an Allocate_Request.submit is received. The source flow allocator determines if the request is well formed. If not well-formed, an Allocate_Response.deliver is invoked with the appropriate error code. If the request is well-formed, a new FlowAllocator-Instance is created and passed the parameters of this Allocate_Request to handle the allocation. It is a matter of DIF policy (AllocateNoificationPolicy) whether an Allocate_Request.deliver is invoked with a status of pending, or whether a response is withheld until an Allocate_Response can be delivered with a status of success or failure.

6.2.2.3 Create_Request Flow

6.2.2.3.1 When invoked

When a flow allocator receives a Create_Request for a Flow object, it means that the requestor is looking for an IPC Process in this DIF that can create a local binding with the Destination_Naming_Info in the request.

6.2.2.3.2 Action upon receipt

When a flow allocator receives a *Create_Request* PDU for a Flow object, it consults its local *DirectoryForwardingTable* to see if it has an entry.

If the address returned by the *DirectoryForwardingTable* is this IPC Process, it creates a FAI and passes the *Create_Request* to it.

If the address returned by the *DirectoryForwardingTable* is not this IPC Process, it is forwarded to the IPC Process indicating result of querying the *DirectoryForwardingTable*.

NOTE Names are granted by a Name Space Manager. Therefore, the Registrar-DAP for that branch of the naming tree will always know where the application is. Other NSM repositories can cache information about a name. The search can progress to “closer” repositories, and if not successful, it is forwarded to the appropriate Registrar-DAP. Hence, there is always at least one entry in the *DirectoryForwardingTable* that is the default result if no other entries are a better match. If the Registrar has not assigned the name, then an error is returned.

6.2.3 Behaviour of a flow allocator instance

6.2.3.1 Allocate_Request.submit

6.2.3.1.1 When invoked

An FAI is created with an *Allocate_Request.submit* when a new Flow Allocation has been requested.

6.2.3.1.2 Action upon receipt

When an FAI is created with an Allocate_Request.submit as input, it will inspect the parameters of the Allocate_Request.

It will then consult its DirectoryForwardingTable to determine where to look for the Destination_ Naming_Info requested.

If the address returned by the DirectoryForwardingTable is on this IPC Process, the requested application is on the same system and a degenerate form of EFCP is invoked to provide local IPC facility within this system.

If the address returned by the DirectoryForwardingTable is not this IPC Process, the FAI will instantiate the required local instances of EFCP and create a A_CREATE CDAP PDU to create a new Flow object, invoking the NewFlowRequestPolicy to send to the IPC Process returned by the lookup in the DirectoryForwardingTable.

"Connection-ids" and the "CurrentConnection-id" attributes of the "Flow" object have to be adequately populated. "Connection-ids" shall contain the ids of all the potential connections that will be required during this flow lifetime. The order of the connection-ids matters, since the ones with a lower index will be used before the ones with a higher index. For each connection-id in the array, the source FAI has to populate the "QoS-cube-id" and "Source-CEP-id" fields (leaving the "Destination-CEP-id" field to the destination FAI). The value of the "CurrentConnection-id" attribute shall be "0".

The FAI sets the Create Request Timer for this CDAP message. The timer indicates the maximum time to wait for a response.

NOTE Some DIFs can have an authoritative DirectoryForwardingTable that can determine whether or not the Application is accessible via this DIF. In those cases, the IPC Process that makes that determination will send a negative A_Create_Response to the originating FAI. For DIFs where this is not the case, the A_Create_Request will time out and after MaxCreateRetries will return an unsuccessful result (negative Create_Response).

6.2.3.2 Allocate_Response.deliver

6.2.3.2.1 When invoked

This API primitive is invoked in response to an Allocate_Request.deliver being delivered to the destination application. The destination application evaluates the request and invokes an Allocate_Response to indicate it accepts or rejects the flow request.

6.2.3.2.2 Action upon receipt

When the FAI gets an Allocate_Response from the destination application, it formulates a *Create_Response* on the flow object requested.

If the response was positive, the FAI will cause EFCP instances to be created to support this allocation. The FAI will iterate the list of Connection-ids in the Flow object and fill in the Destination-CEP-id attribute. A positive Create_Response Flow is sent to the requesting FAI with the connection-ids and other information provided by the destination FAI.

The Create_Response is sent to requesting FAI with the necessary information reflecting the existing flow, or an indication as to why the flow was refused. If the response was negative, the FAI does any necessary housekeeping and terminates.

6.2.3.3 Create_Request PDU

6.2.3.3.1 When invoked

This is provided as input to the destination FAI to create the other end of a flow.

6.2.3.3.2 Action upon receipt

When a FAI is created with a Create_Request (Flow) as input, it will inspect the parameters first to determine if the requesting Application (Source_Naming_Info) has access to the requested Application (Destination_Naming_Info) by inspecting the Access Control parameter.

If not, a negative Create_Response PDU will be returned to the requesting FAI.

If it does have access, the FAI will determine if the policies proposed are acceptable, invoking the NewFlowRequestPolicy. If not, a negative Create_Response PDU is sent.

If they are acceptable, the FAI will invoke an Allocate_Request.deliver primitive to notify the requested Application that it has an outstanding allocation request. If the application is not executing, the FAI will cause the application to be instantiated.

6.2.3.4 Create_Response PDU

6.2.3.4.1 When invoked

This PDU is received when the destination FAI responds to a Create_Request PDU.

6.2.3.4.2 Action upon receipt

When a Create_Response PDU is received, its InvokeID is used to deliver it to the appropriate FAI.

If the response is negative, the Allocation invokes the AllocateRetryPolicy. If the AllocateRetryPolicy returns a positive result, a new Create_Request PDU is sent and the CreateFlowTimer is reset.

Otherwise, if the AllocateRetryPolicy returns a negative result or the MaxCreateRetries has been exceeded, an Allocate_Request.deliver primitive is returned to notify the Application that the flow could not be created. If the reason was "Application Not Found," the primitive may be referred to the DIF-Allocator to search further.

The FAI deletes the EFCP instances it created and does any other clean-up necessary, before terminating.

If the response is positive, it completes the binding of the EFCP with this connection-endpoint-id to port-id of the requesting Application and invokes a Allocate_Response.deliver primitive to notify the requesting Application that its allocation request has been satisfied.

6.2.3.5 Deallocate (port-id)

6.2.3.5.1 When invoked

This API call is made by an application, which has allocated a flow to notify the DIF that it will deallocate the flow (port-id).

6.2.3.5.2 Action upon receipt

When a deallocate primitive is invoked, it is passed to the FAI responsible for that port-id. The FAI sends a Delete Request CDAP PDU on the Flow Object referencing the destination port-id, deletes the local binding between the Application and the DTI and either completes any housekeeping and terminates or waits for the response (it depends on the *A_DELETE* Request requiring a response or not, which is optional and decided by the FAI).

NOTE The EFCP instances will be deleted automatically after 2 or 3Δt.

6.2.3.6 Delete_Request flow object

6.2.3.6.1 When invoked

This CDAP PDU notifies the peer FAI that the Flow object is being deleted.

6.2.3.6.2 Action upon receipt

When this CDAP PDU is received by the FAI identified by the port-id carried in the PDU, the FAI invokes a Deallocate.deliver to notify the local Application, deletes the binding between the Application and the

local EFCP, and, if required by the Delete_Request, sends a Delete_Response PDU indicating the result and terminates after 2 or 3Δt.

6.2.3.7 Delete_Response flow object

6.2.3.7.1 When invoked

This PDU is received in response to a Delete_Request.

6.2.3.7.2 Action upon receipt

When a Delete_Response PDU is received, the FAI completes any clean-up and terminates (in the case the FAI had requested a Delete_Response).

6.2.4 Events

6.2.4.1 General

Once the flow is created, the FAI may monitor the flow for significant events. The action for required or common events is described in this subclause. A specific DIF may define other events that the FAI should be aware of.

6.2.4.2 Create_Request Timer

6.2.4.2.1 When invoked

This timer is set when a Create_Request PDU is sent to find a requested destination application. If it expires, it indicates that either the Create_Request PDU was lost or the application was not found.

6.2.4.2.2 Action on expiration

When this timer expires and CreateRetries is less than MaxCreateRetries, the FAI re-sends the Create_Request PDU.

If CreateRetries is greater than or equal to MaxCreateRetries, the FAI invokes a negative Allocate_Response.deliver notifying the Application that its request has failed, and does any clean-up before terminating.

6.2.4.3 Change of address

6.2.4.3.1 When invoked

This event occurs when the Resource Allocator has determined (or been told) that the address of this IPC-Process shall be changed.

It is important that the Resource Allocator has acted on this event prior to notifying the flow allocator, so that routing advertisements of the new address will have started. The old address remains in effect and PDUs received with the old address are considered valid. The old address is allowed to remain in the system until it is replaced, i.e. with no updates containing the old address it is eventually deleted from the routing tables of the IPCPs of this DIF. When the old address is deleted, the FAI should be notified.

6.2.4.3.2 Action upon receipt

IF the Application-Process-Instance-id is this IPCP

CurrentAddress = Previous_Address from the Write_Request THEN