INTERNATIONAL STANDARD

ISO/IEC 14752

First edition
2000-04-15

Information technology — Open Distributed Processing — Protocol support for computational interactions

Technologies de l'information — Traitement distribué ouvert — Support du protocole pour les interactions d'ordinateurs

Reference number
ISO/IEC 14752:2000(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

CONTENTS

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14752 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software engineering*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.931.

Annex A forms a normative part of this International Standard. Annex B is for information only.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN DISTRIBUTED PROCESSING – PROTOCOL SUPPORT FOR COMPUTATIONAL INTERACTIONS

1 Scope

This Recommendation | International Standard is based on the framework of abstractions and concepts developed in the Reference Model for Open Distributed Processing (ITU-T Rec. X.902 | ISO/IEC 10746-2 and ITU-T Rec. X.903 | ISO/IEC 10746-3).

This Recommendation | International Standard defines how interactions between computational objects in a computational specification of a system relate to protocol support for those interactions in an engineering specification of that system. In particular it:

- defines a General Interworking Framework (GIF);
- within the GIF, defines a set of facilities each comprising a set of functionally-related service primitives as abstract definitions of the interactions of basic engineering objects and channel objects;
- defines the parameters of the service primitives of the GIF;
- defines the permitted sequence of the service primitives by means of state tables;
- specifies, in annexes, the mapping of the GIF service primitives and their parameters to the messages and fields of particular protocols.

As specified in this Recommendation | International Standard, the GIF defines protocol support for a pragmatic subset of the possible computational interactions defined in ITU-T Rec. X.903 | ISO/IEC 10746-3. It is also restricted in the features of the protocol support and the supported transparencies.

The GIF, as specified here, defines:

- support for computational operations, but not for streams;
- support using stub, binder and protocol objects hierarchically, such that any interaction at the interworking reference point of the supporting protocol object supports liaisons of one of those objects or of the basic engineering object, and any interaction to support those liaisons is passed via that interworking reference point; and
- interactions at a single interworking reference point, from the perspective of one side; interceptors are not explicitly considered;

NOTE 1 – It is intended that the GIF could be extended, in a future amendment, to support streams and flows. The present specification is restricted to areas that are technically stable.

The GIF supports at least some forms of:

- access transparency; and
- location transparency.

The GIF as specified here also supports a limited equivalent of relocation transparency. Other transparencies are not addressed in this present specification.

NOTE 2 – It is intended that the GIF could be extended, in future amendments, to support additional transparencies.

The GIF does not explicitly model Quality of Service requirements.

The application of security-related issues to the GIF are not included in the current text and are for further study.

The set of mappings to particular protocols specified in annexes to this Recommendation | International Standard is not exhaustive. The GIF could be mapped to other protocols.

NOTE 3 – In particular, a mapping to the DCOM protocol family would be a candidate for an additional annex.

2 Normative References

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendation | International Standards

- ITU-T Recommendation X.210 (1993) | ISO/IEC 10731:1994, *Information technology – Open systems interconnection – Basic Reference Model – Conventions for the definition of OSI services*.
- ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1996, *Information technology – Open distributed processing – Reference Model: Foundations*.
- ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1996, *Information technology – Open distributed processing – Reference Model: Architecture*.
- ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1999, *Information technology – Open distributed processing – Interface definition language*.
- ITU-T Recommendation X.930 (1998) | ISO/IEC 14753:1999, *Information technology – Open distributed processing – Interface references and bindings*.

2.2 Other Specifications

The edition of [CORBA 2] indicated below was valid at the time of publication of this Recommendation | International Standard. [CORBA 2] is subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying later editions of [CORBA 2] when they become available.

- [CORBA 2] – *The Common Object Request Broker: Architecture and Specification, Revision 2.3, Object Management Group, December 1998 (OMG Doc Number: Formal/98-12-01).*
- RFC 793, "Transmission Control Protocol", 1981.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 Terms defined in the ODP Reference Model: Foundations

This Specification makes use of the following terms defined in ITU-T Rec. X.902 | ISO/IEC 10746-2:

a) binding;

c) client object;

e) initiating object;

d) interface;

e) interface signature;

f) name;

g) object;

h) reference point;

i) responding object;

j) server object;

k) viewpoint.

3.2 Terms defined in the ODP Reference Model: Architecture

This Specification makes use of the following terms defined in ITU-T Rec. X.903 | ISO/IEC 10746-3.

a) announcement;
b) basic engineering object;
c) binder;
d) capsule
e) channel;
f) computational object;
g) computational language;
h) computational viewpoint;
i) engineering viewpoint;
j) interrogation;
k) interceptor;
l) invocation;
m) (computational) operation;
n) operation interface;
o) protocol object;
p) signal;
q) signal interface;
r) stub;
s) termination.

3.3 Definitions for protocol support for computational interactions

This Specification makes use of the following terms.

3.3.1 access facility: A set of service primitives that allow a stub objects to negotiate the abstract and transfer syntax to be used for the operation data to be transmitted over the channel.

3.3.2 association: A relationship (binding) between protocol objects (or between a protocol object and an interceptor) that is established independently of the protocol exchanges that support a particular computational interaction.

3.3.3 association management facility: A set of service primitives which support the management of an association between protocol objects.

3.3.4 basic interworking facility: A set of service primitives which have a direct correspondence with computational signals which model computational operations.

3.3.5 client-side: A node, cluster or capsule, which:

a) contains a basic engineering object corresponding to a computational client object; and
b) contains, or is potentially capable of containing, stub, binder and protocol objects in a channel supporting operations involving the client object.

The term client-side is used prior to the establishment of a channel, during the channel's lifetime and after it has terminated.

3.3.6 deliver primitive: A service primitive for which the protocol object is the responding object of the corresponding communication.

3.3.7 invocation submit: A signal in the implicitly defined signal interface of a client computational object which has the same name and parameters as the invocation of an interrogation or announcement in the original operation interface.

3.3.8 **invocation deliver**: A signal in the implicitly defined signal interface of a server computational object which has the same name and parameters as the invocation of an interrogation or announcement in the original operation interface.

3.3.9 **location facility**: A set of service primitives that allow a client-side binder object to ask a server-side if it will accept requests carrying invocations to a particular (computational) server object. The server-side can confirm or reject the proposal or suggest an alternative server-side that is capable of handling requests.

3.3.10 **server-side**: A node, cluster or capsule, which:

a) contains, or is potentially capable of containing, a basic engineering object that corresponds to a computational server object and stub, binder and protocol objects in a channel supporting operations involving the server object; or

b) contains, or is a potentially capable of containing, a protocol object which (possibly via interactions with other engineering objects) can return a reply identifying another server-side.

The term server-side is used prior to the establishment of a channel, during the channel's lifetime and after it has terminated. It is also used when an appropriate basic engineering object cannot be instantiated following some received message.

3.3.11 **service primitive**: An abstract definition of an interaction of channel objects that causes protocol exchanges between the protocol objects in the channel.

3.3.12 **submit primitive**: A service primitive for which the protocol object is the initiating object of the corresponding communication.

3.3.13 **termination deliver**: A signal in the implicitly defined signal interface of a client computational object which has the same name and parameters as one of the terminations of an interrogation in the original operation interface.

3.3.14 **termination submit**: A signal in the implicitly defined signal interface of a server computational object which has the same name and parameters as one of the terminations of an interrogation in the original operation interface.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply.

GIF	General Interworking Framework
ODP	Open Distributed Processing
ODP IDL	Open Distributed Processing Interface Definition Language
OSI	Open Systems Interconnection
psci	Protocol Support for Computational Interactions
RM-ODP	Open Distributed Processing: Reference Model
TCP	Transmission Control Protocol

5 Conventions

State tables are used to specify the allowed sequence of primitives in each of the facilities of the GIF. Each state machine is initially in the "idle" state. A particular primitive is only permitted if the intersection of the current state and that primitive is non-blank. The entry in the cell defines the state subsequent to the primitive The states are defined by the top row of the state table and have self-explanatory names. If there is an inconsistency between the natural language description of state transitions and the corresponding state table, the natural language description shall take precedence.

6 Overview

6.1 General Interworking Framework

As defined in ITU-T Rec. X.903 | ISO/IEC 10746-3, operations in the computational viewpoint correspond in the engineering viewpoint to interactions between basic engineering objects. Where these engineering interactions in the engineering viewpoint are distributed, a channel connects the basic engineering objects. The establishment and use of the channel involves interactions between the various kinds of engineering object in the channel. In some form or other,

these interactions result in observable events occurring at the interworking interface of the protocol objects, according to the rules of one or more protocol specifications. At least some of these observable events will have a direct correspondence to the computational interactions. Other protocol events will be concerned with the management of the engineering binding including the support of required transparencies.

The General Interworking Framework (GIF) defined in this Recommendation | International Standard defines an abstraction of the engineering interactions of the basic engineering and channel objects, including the correspondence between computational operations and the relevant engineering interactions. Mappings of the GIF to particular, implementable, protocols are specified in annexes to this Recommendation | International Standard.

The GIF comprises a set of *facilities* each consisting of a number of *service primitives*. Each facility supports the liaison between one kind of channel object (e.g. stub, binder, protocol) and the service primitives are abstract definitions of the interactions of that kind of object with its peers.

NOTE 1 – If the channel objects were themselves considered as computational objects, a facility would be seen as an interface, and the service primitives as signals.

A mapping of the GIF to a particular protocol will typically specify constraints on the sequencing of the service primitives between different facilities. Such sequencing constraints are not included in the GIF which is intended to support mappings to protocols with a variety of properties and capabilities.

NOTE 2 – For example, mappings to protocols using non-blocking connections, blocking connections and connectionless protocols each give rise to different constraints.

GIF supports both evolution and extensibility. Evolution is obtained by defining, in the abstract form of the service primitives, the architecture used and the messages exchanged for communication. Extensibility is offered by introducing optionality of some of the service primitives and by flexibility in the mapping from service primitives to particular protocols.

NOTE 3 – The GIF itself could also be extended, adding further service primitives or facilities. Such extensions could address additional transparencies.

6.2 Liaisons between channel objects

A distributed engineering binding between basic engineering objects, corresponding to a computational binding is supported by liaisons between the peer channel objects on the client and server sides. Figure 1 shows the relationships between the various objects.

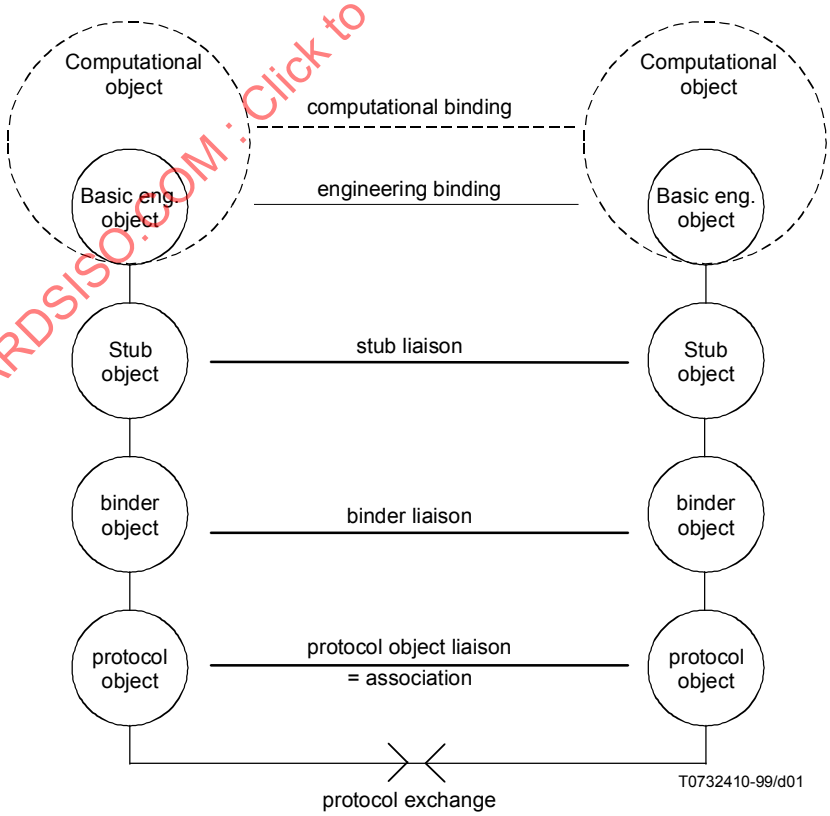

Figure 1 – Relationships of channel objects

In Figure 1, it should be noted that the computational and basic engineering objects are just corresponding views of the same thing. There is no implication that one is contained in the other although other basic engineering objects may correspond to the same computational object (see 10.2 of ITU-T Rec. X.903 | ISO/IEC 10746-3). Similarly, the computational binding and the engineering binding are different views of the same thing.

The hierarchical ordering of the channel objects and of their liaisons in Figure 1 represents the static position when the liaisons are complete and are supporting a particular instance of a computational interaction. The hierarchy should not be taken as implying restrictions on when the liaisons are established or how the channel objects interact during establishment or other liaison management. Establishment of the various liaisons may involve interaction between any of the channel objects in a given node. Other engineering objects may also be involved, in some cases causing protocol exchanges at the interworking reference point. Liaison management, including establishment, may also use other paths than the one being managed. Establishment of any of the liaisons in Figure 1 can take place at an earlier epoch, or can be overlapped with the establishment of the other liaisons. Where recovery of failed bindings is supported, the establishment of a replacement channel may involve the establishment of new liaisons (to the same or different channel objects) or the modification of the surviving liaisons.

For a particular protocol, a single event at the interworking reference point may carry semantics from several of the liaisons. This is termed "piggy-backing".

The stub, binder and protocol liaisons can be established independently of the support of a single instance of a computational interaction. These liaisons can be:

1) established prior to any specific computational interaction;
2) used for interactions between a number of different computational objects; and
3) used for an indefinite number of computational interactions, either consecutively or concurrently.

The stub, binder and protocol liaisons can also be transient, with the duration of the shared context between peer channel objects being limited to the support of a single computational interaction.

NOTE – For example, a server-side object could maintain the state that supports a liaison only from the receipt of a request to the issue of the reply.

6.3 Facilities of the GIF

As stated above, the GIF defines a set of facilities, each comprising a number of service primitives which are functionally related. Each facility is primarily the concern of one kind of engineering object.

The **basic interworking facility** supports the liaison of the basic engineering objects. It includes service primitives which have a direct correspondence with the signals which model the computational operations. This facility is supported by all protocols that support computational operations.

The **access facility** supports access transparency and is primarily the concern of the stub objects. The primitives concern the negotiation of the representation of data to be transmitted via the channel.

The **location facility** supports location transparency and is primarily the concern of the binder objects. It includes service primitives that allow a client-side protocol object to ask a server-side protocol object if it is an appropriate destination for access to a particular basic engineering object, and to allow a server-side protocol object to propose some other server-side. The use of this facility can be combined with the basic interworking facility to allow a server-side, where the client-side "expected" the target basic engineering object to be, to propose an alternative server-side. This allows a limited equivalent of relocation transparency.

The **association management facility** defines service primitives that manage associations - liaisons between protocol objects. An association has an existence independent of a particular instance of a computational interaction (see 8.2).

6.4 Computational operations and signals

The RM-ODP introduces the concept of signal to express the semantics of more abstract interactions. A signal is atomic and localised at a reference point, and represents the initiation or completion at that reference point of some more abstract distributed interaction. Thus an announcement can be delimited by two signals, representing its initiation at a reference point in the sending system and its delivery at a reference point in the receiving system. Similarly, an interrogation can be delimited by two pairs of signals.

A computational operation, as defined in the computational language description in the RM-ODP, is an interaction between a client object and a server object. It can be either an interrogation, consisting of two interactions - an invocation from the client and a replying termination, or be an announcement from the client to the server, with just an invocation. The components of the computational operations - invocation and termination - are characterised as resulting in the conveyance of information from one object to another (see ITU-T Rec. X.903 | ISO/IEC 10746-3).

In defining the protocol support for computational operations it is useful to model the operations as signals (as described in ITU-T Rec. X.903 | ISO/IEC 10746-3). For the purposes of this Recommendation | International Standard, it is assumed that for any defined operation interface there are implicitly defined signal interfaces, one for the client and one for the server, such that:

- for each invocation in the operation interface there is an **invocation submit** signal in the signal interface at the client and an **invocation deliver** signal in the signal interface at the server;
- for each termination in an (interrogation) operation interface there is a **termination submit** in the signal interface at the server and a **termination deliver** signal in the signal interface at the client.

Each signal has the same name and parameters as the corresponding component of the operation.

For an announcement operation, the invocation is mapped to invocation submit and invocation deliver signals. Within the GIF, there is no form of reply or acknowledgement to an announcement.

NOTE – A particular protocol may or may not provide Quality of Service mechanisms that involve acknowledgement of announcements.

6.5 Encoding of computational information

The encoding and decoding of the information to be conveyed between the computational objects, the names of the invocation and terminations and their parameters (which are also the names and parameters of the corresponding signals) are considered to be performed by stub objects. The encoding depends on the protocol that is used, but can be:

a) determined simply by the protocol specification, with or without options chosen by the sender; or

b) one of a set of encodings, constrained such that a receiver can determine which is chosen, with the choice made by the sender; or

c) subject to negotiation on the association, prior to the interactions that correspond to the computational operation; or

d) determined by previous exchanges with the expected receiver, though not necessarily on the association; or

e) dependent on the expected receiver, with the details as to which encoding to use made available to the sender by some indirect means (i.e. not from previous exchanges directly with the receiver).

7 Interface references

Interface references are introduced in 8.2.2 of ITU-T Rec. X.903 | ISO/IEC 10746-3 and are further refined in Interface references and binding (ITU-T Rec. X.930 | ISO/IEC 14753).

NOTE – The specification of the mapping of GIF to a particular protocol will typically include the specification of the interface reference format.

8 Service model

8.1 Service primitives

The primitives defined in clauses 9 to 12 are abstract definitions of interactions of engineering objects defined as existing in a channel. The service primitives are abstract definitions of interactions of the channel objects that cause protocol exchanges between the protocol objects.

The mapping of the GIF to a particular protocol will specify that at least some of the service primitives correspond to the sending or receiving of protocol messages at the interworking interface of the (underlying) protocol object. The mapping may involve "piggy-backing", where more than one service primitive (often, but not always, from the same or different facilities) corresponds to the protocol interaction.

Primitives are classified as either:

a) submit primitives, where the protocol object is the initiating object of the corresponding communication (i.e. protocol elements are sent from this side's protocol object); or

b) deliver primitives, where the protocol object is the responding object of the corresponding communication (i.e. protocol elements are received by this side's protocol object).

For all submit primitives, there is a corresponding deliver primitive which will (typically) have the same parameters. Following the occurrence of a submit primitive at some protocol object, the underlying communication mechanisms cause one of the following:

a) successful communication - the corresponding deliver primitive occurs at the appropriate responding protocol object with identical values for all the matching parameters; or

b) reported unsuccessful communication - a different deliver primitive, occurs at the original protocol object and the type and parameters of the deliver primitive imply that a) will not occur; or

c) unreported unsuccessful communication - no specific deliver primitive at either protocol object.

Case c) includes all cases where the non-occurrence of a) is not reported to the initiating object.

NOTE – Possible events occurring under case c) include the corresponding deliver primitives occurring at the intended responding object but with different (erroneous) parameters, error reporting deliver primitives at other interfaces or arbitrary deliver primitives at any interface.

There are some deliver primitives for which there is no corresponding submit primitive. These represent the signalling to the protocol object of conditions occurring autonomously in the underlying communications.

Where the interworking interface of a protocol object interacts with another protocol object via an interceptor object, any transformation performed by the interceptor does not affect the interaction described by the service primitives.

Each primitive has a number of parameters. The number, names and purpose of these parameters are defined below.

Quality of Service requirements can be associated with any of the service primitives. The QoS requirements are not modelled as parameters, but a particular mapping or implementation could treat them as additional parameters. Equally, they could be determined via other means or interfaces.

8.2 Associations

Protocol objects can exchange protocol to establish bindings that exist independently of the support of a single instance of a computational interaction. Such a binding is called an **association**. An association can be:

1) established prior to any specific computational interaction;

2) used for interactions between a number of different computational objects; and

3) used for an indefinite number of computational interactions, either consecutively or concurrently.

NOTE 1 – An association will often correspond to a "connection" in the terminology of the supporting protocol. However, an association can also be supported by a series of messages exchanged over a connectionless protocol.

An instance of an interworking interface that has established an association can be referred to as an **association endpoint**. Any service primitive occurring at that endpoint occurs in the context of the association.

Establishment of the stub and binder liaisons may also cause protocol exchanges on the association, including during association establishment, prior to the support of a particular computational interaction.

In general, there is no necessary relationship between establishment of an association and quality of service. However, a specific protocol can establish associations that are deemed to be **reliable**. For the purposes of this Specification, a reliable association is one which it is assumed that following a submit primitive at either association endpoint the underlying communications will ensure that:

a) the corresponding deliver primitive, with identical values for the parameters, occurs at the other association endpoint; or

b) a deliver primitive indicating an error occurs at one or other of the association endpoints or (possibly different) primitives indicating errors occur at both endpoints; or

c) deliver primitives terminating the association are delivered at both endpoints.

The underlying communications supporting a reliable association are also assumed to ensure that a sequence of submit primitives at one endpoint produce the corresponding sequence of deliver primitives at the other endpoint, unless an error occurs.

NOTE 2 – Quality of Service service issues, which pertain, among other things, to the validity of the assumptions of reliability, are matters of separate standardization.

9 Basic interworking facility

The basic interworking facility defines primitives that directly support computational interactions. The service primitives of the basic interworking facility are used directly by the basic engineering object. Some of these service primitives correspond directly to the computational signals (including the computational signals that are mapped from the components of the computational operations).

Corresponding pairs of submit and deliver primitives are grouped together, since they share the same parameters.

9.1 Request

9.1.1 Purpose

The request submit and request deliver primitives carry an operation invocation from a client-side basic engineering object to a server-side basic engineering object.

9.1.2 Request submit

The request submit primitive occurs at a basic engineering object corresponding to a client computational object.

The request submit primitive corresponds directly to an **invocation submit** signal in a signal interface corresponding to a client operation interface.

The occurrence and parameters of the request submit primitive depend on the occurrence and parameters of the invocation submit signal at the client object interface.

9.1.3 Request deliver

The request deliver primitive occurs at a basic engineering object supporting a server computational object.

The request deliver primitive corresponds directly to an **invocation deliver** signal in a signal interface corresponding to a server operation interface.

If the server object interface identified by the parameters of the request deliver primitive is instantiated at the server-side, an invocation deliver signal occurs at the server object interface with parameters determined by the parameters of the request deliver primitive.

NOTE – "is instantiated" includes the case of the instantiation of the server object as a result of the request deliver.

If the server object interface identified by the parameters of the request deliver primitive is not instantiated at the server, a result submit may be issued by the server.

9.1.4 Parameters

The parameters of the request submit and request deliver primitive are identical. They are:

operation name: the name of the corresponding computational operation. This parameter is always present.

target interface reference: an Interface Reference that identifies the basic engineering object that corresponds to the interface of the computational server object. This parameter is always present.

invocation type: identifies the type of the corresponding computational operation. It has one of the following values:

- interrogation;
- announcement.

This parameter is always present.

NOTE 1 – The value of the parameter can be logically derived from the interface type contained in the interface reference, but the parameter is distinguished for clarity.

operation parameters: the values of the actual parameters corresponding to the formal parameters in the invocation signature of the computational operation. This parameter shall be present unless there are no parameters in the invocation signature.

request reference: an identification of this request that can be used by subsequent primitives to unambiguously refer to this request among all request primitives. The request reference shall remain unambiguous until the occurrence of a primitive that specifies the reference is no longer needed. This parameter shall be present if the invocation type is "interrogation".

NOTE 2 – Service primitives of several different facilities have request references. In the GIF, the request references of each facility are considered distinct. In the mapping to a particular protocol, a single set of identifiers may be used for the primitives of more than one facility. In such a case, the requirement for unambiguity covers all the relevant primitives.

NOTE 3 – In general, the request reference is globally unambiguous, but if the specific protocol establishes an association and the mapping is such that a result is always returned on the association on which the request was sent, any transmitted value for the request reference need only be unambiguous in the scope of that association. If only one request is outstanding on a association at a time, the request reference can be entirely implicit in the end-points of the association and no value is transmitted in the protocol.

9.2 Result

9.2.1 Purpose

The result submit and result deliver primitives carry an operation termination from a server-side basic engineering object to a client-side basic engineering object.

9.2.2 Result submit

The result submit primitive occurs at a basic engineering object supporting a server object.

The result submit primitive corresponds directly to an **termination submit** signal in a signal interface corresponding to a server operation interface, where the operation is an interrogation. In this case, the occurrence and parameters of the result submit primitive at the basic engineering object depend on the occurrence and parameters of the termination submit signal at the server object interface.

9.2.3 Result deliver

The result deliver primitive occurs at a basic engineering object supporting a client object.

The result deliver primitive corresponds directly to an **termination deliver** signal in a signal interface corresponding to a client operation interface, where the operation is an interrogation. The name and parameters of the termination deliver signal are determined by the parameters of the result deliver primitive.

NOTE – A failure of the underlying communications including a failure to communicate with the target engineering object, may or may not map to a termination deliver signal. Whether it does depends on the specification of the computational operation and on the language binding.

9.2.4 Parameters

The parameters of the result submit and result deliver primitives are identical. They are:

request reference: the request reference of the request primitive that corresponded to the invocation of the computational operation to which this is the termination.

termination name: the name of the termination operation. This parameter is present if, and only if, the termination is named.

termination parameters: the values of the actual parameters corresponding to the formal parameters of the termination signature identified by the operation name.

9.2.5 Effects

Following the result primitive, the request reference is no longer needed.

9.3 Cancel

9.3.1 Purpose

The cancel primitive is used by a client side to inform the server side that it no longer requires a reply to an earlier request primitive.

9.3.2 Cancel submit

The cancel submit primitive occurs at a client-side basic engineering object which has issued a request submit primitive and has not subsequently received a result deliver with a matching request reference parameter.

The client side may receive a result deliver with the matching request reference after issuing a cancel submit.

9.3.3 Cancel deliver

The cancel deliver primitive occurs at a server-side basic engineering object.

The cancel deliver primitive is advisory. The server side can, but need not issue a result submit with the same request reference after receiving a cancel deliver. The client side may receive a result deliver after issuing a cancel submit primitive.

9.3.4 Parameters

The parameters of the cancel primitive is:

request reference: the request reference of the request submit primitive that is being cancelled.

The request reference does NOT become "no longer needed" after the cancel primitive is used.

9.4 Abort

9.4.1 Purpose

The abort primitive is a deliver-only primitive which signals to the basic engineering object that a failure of the underlying communications has occurred and a particular request/reply pair will not be completed. On a client-side, no result deliver primitive will occur for some previous request submit. On a server-side, no result submit will be accepted for an outstanding request deliver.

9.4.2 Abort deliver

The abort deliver primitive occurs at a client or server-side basic engineering object.

9.4.3 Parameters

The parameter of the abort primitive is:

request reference: the request reference of the request primitive that was pending.

9.4.4 Effects

The request reference becomes "no longer needed" after the abort deliver primitive.

9.5 State table for the Basic Interworking Facility

The Basic Interworking Facility is asymmetric between client and server-side. For each side, a separate state machine exists for each request reference. Primitives are assigned to the state machine that has the corresponding request reference.

9.5.1 Client-side state table

Table 1 – Client-side state table for basic interworking facility

State / Incoming event	Idle	wait-reply	cancelled
request submit	wait-reply		
result deliver		idle	idle
cancel submit		cancelled	
abort deliver	idle	idle	idle
NOTE – This state machine, which relates to a single request reference can remain in the cancelled state indefinitely. This reflects the fact that the cancel submit does not make the request reference "no longer needed".			

9.5.2 Server-side state table

Table 2 – Server-side state table for basic interworking facility

State / Incoming event	Idle	wait-reply
request deliver	wait-reply	
result submit		idle
cancel deliver	idle	idle
abort deliver	idle	idle

10 Access facility

The access facility is used by the stub objects to negotiate the abstract and transfer syntax to be used for the operation data to be transmitted over the channel. One side proposes a set of syntaxes that would be suitable. The other side chooses a subset of those offered. Subsequent messages (arising from the request submit and result submit primitives of the basic interworking facility) are then parameterised within the accepted subset to identify which syntax is being used.

A particular protocol may define the cardinality of accepted subset to be (at most) one, in which case the subsequent parameterisation is implicit.

The access facility is symmetric with respect to the client/server polarity of the channel. In the GIF, either side may propose a set of syntaxes to be used.

NOTE – A particular protocol may support only one direction.

The access facility is not required to be supported by all protocols. If it is not supported, the stub liaison is established by other means.

10.1 Syntax-propose

10.1.1 Purpose

The syntax-propose is used by a stub object to propose a set of syntaxes (i.e. abstract-transfer syntax pairs) for the encoding of subsequent messages.

10.1.2 Syntax-propose submit

The syntax-propose submit primitive occurs at a stub object supporting a client or a server object.

10.1.3 Syntax-propose deliver

The syntax-propose deliver primitive occurs at a stub object supporting a client or server object.

10.1.4 Parameters

The parameters of the syntax-propose primitives are:

offered syntax set: A set of abstract and transfer syntax pairs (The same abstract syntax may be offered with more than one transfer syntax). This parameter is always present.

request reference: An identification of this syntax-propose that can be used by subsequent primitives (especially the syntax-advise) to unambiguously refer to this syntax-propose among all syntax-propose primitives. The request reference shall remain unambiguous until the occurrence of a primitive that specifies the reference is no longer needed. This parameter is always present.

NOTE 1 – Service primitives of several different facilities have request references. In the GIF, the request references of each facility are considered distinct. In the mapping to a particular protocol, a single set of identifiers may be used for the primitives of more than one facility. In such a case, the requirement for unambiguity covers all the relevant primitives.

NOTE 2 – In general, the request reference is globally unambiguous, but if the specific protocol establishes an association, any transmitted value for the request reference need only be unambiguous in the scope of that association. If only one primitive defined as having a request reference is outstanding on an association at a time, the request reference can be entirely implicit in the end-points of the association and no value is transmitted in the protocol.

10.2 Syntax-advise

10.2.1 Purpose

The syntax-advise primitives are used by a stub object to reply to a syntax-propose primitive, selecting a subset of the offered syntax pairs.

10.2.2 Syntax-advise submit

The syntax-advise submit primitive occurs at a stub object subsequent to a syntax-propose deliver primitive.

10.2.3 Syntax-advise deliver

The syntax-advise deliver primitive occurs at a stub object subsequent to a syntax-propose submit primitive.

10.2.4 Parameters

The parameters of the syntax-advise primitive are:

request reference: the request reference of the syntax-propose primitive to which this is the reply.

Accepted syntax set: a subset of the abstract and transfer syntax pairs in the offered syntax set of the corresponding syntax-propose primitive. This parameter is always present.

10.2.5 Effects

Following the syntax-advise primitive, the request reference is no longer needed.

If the accepted syntax set is empty, syntax negotiation has failed. Subsequent behaviour is not specified in the GIF.

If the accepted syntax set contains precisely one syntax pair, those syntaxes shall be used for the marshalling and encoding of subsequent messages sent as a result of Request and Result submit primitives. The stub object receiving a Request or Result deliver primitive can assume that they have been encoded using the syntax pair in the accepted syntax set.

If the accepted syntax set contains more than one syntax pair, the sender of each subsequent message sent as a result of a Request or Result submit primitive shall determine which syntaxes to use for marshalling and encoding. This need not be the same for the whole message. The message (or the parts of the message) shall be parameterised to identify which syntax pair is used. The stub object receiving a Request or Result deliver primitive will use the parameter to determine the correct decoding and unmarshalling.

NOTE – A particular protocol may place various constraints or simplifications on which syntax is used for which message.

10.3 Access-cancel

10.3.1 Purpose

The access-cancel primitive is used by the sender of a syntax-propose to inform the other side that it no longer requires the stub liaison.

10.3.2 Access-cancel submit

The access-cancel submit primitive occurs at a stub object which has issued a syntax-propose primitive and has not subsequently received a syntax-advise deliver with a matching request reference parameter.

The client side may receive a syntax-advise deliver with the matching request reference after issuing a access-cancel submit.

10.3.3 Access-cancel deliver

The access-cancel deliver primitive occurs at a stub object.

If the stub object has already issued an access-advice submit with the matching request reference, the access-cancel deliver has no effect and can be ignored. If no access-advice submit has been issued, the stub liaison is terminated.

10.3.4 Parameters

The parameters of the access-cancel primitive is:

request reference: the request reference of the syntax-propose primitive that is being cancelled.

The request reference does NOT become "no longer needed" after the access-cancel primitive is used.

10.4 Access-abort

10.4.1 Purpose

The access-abort submit and deliver primitives terminate the stub liaison or an attempt to establish a stub liaison. They can be issued or received during attempted establishment of the stub liaison or after establishment is complete. The access-abort deliver primitive can also occur as a result of events in the supporting communications.

10.4.2 Access-abort submit

The access-abort submit primitive occurs at a client or server-side stub object.

10.4.3 Access-abort deliver

The access-abort deliver primitive occurs at a client or server-side stub object.

The primitive can occur as a consequence of an access-abort submit primitive issued by the other side, or as a consequence of events in the underlying communications.

10.4.4 Parameters

The parameter of the access-abort primitives is only present if the primitive is issued or received during liaison establishment:

request reference: the request reference of the syntax-propose primitive that was pending.

If the stub liaison is established (i.e. there is no outstanding syntax-propose), there are no parameters on the access-abort deliver primitive.

10.4.5 Effects

The request reference, if present, become "no longer needed" after the access-abort deliver primitive.

The stub liaison is terminated.

10.5 State table for the Access Facility

The Access Facility is symmetric between client-side and server-side and a single state table is used for both sides. The "our-req pending" and "their-req pending" refer to whether the pending request was issued by this side or the other side. The "data transfer" event refers to any primitive of the Basic Interworking Facility except the abort deliver.

Table 3 – State table for access facility

State / Incoming event	Idle	our-req pending	their-req pending	available	cancelled
syntax-propose submit	our-req pending				
syntax-propose deliver	their-req pending				
syntax-advise submit			available		
syntax-advise deliver		available			available
data transfer submit				available	
data transfer deliver				available	
access-cancel submit		cancelled			
access-cancel deliver			idle	available	
access-abort submit		idle	idle	idle	idle
access-abort deliver		idle	idle	idle	idle

Although the Access Facility is symmetric between the sides, any protocol mapping is required to ensure that collisions do not occur.

NOTE – A mapping that treats an apparent collision as establishing two stub liaisons both using the same underlying association is considered to ensure that collisions do not occur.

11 Location facility

The Location facility comprises a set of primitives that allow a client-side binder object to ask a server-side if it will accept requests carrying invocations to a particular (computational) server object. The server-side can confirm or reject the proposal or suggest an alternative server-side that is capable of handling requests.

11.1 Location-query

11.1.1 Purpose

The location-query is used by the client-side to ask a server side if the server-side is the location for a basic engineering object (corresponding to a computational object).

11.1.2 Location-query submit

The location-query submit primitive occurs at a binder object supporting a client object.

11.1.3 Location-query deliver

The location-query deliver primitive occurs at a server-side binder object.

NOTE – Although the location-query deliver primitive is considered to occur at a server-side, it is not required that the binder object gives access to any basic engineering object corresponding to a computational server object. The binder object may exist only to redirect clients to an appropriate server. Such redirection may be temporary or permanent.

11.1.4 Parameters

The parameters of the location-query primitives are:

target interface reference: An Interface Reference that identifies the basic engineering object whose location is sought. This parameter is always present.

request reference: An identification of this location-query that can be used by subsequent primitives to unambiguously refer to this location-query among all location-query primitives. The request reference shall remain unambiguous until the occurrence of a primitive that specifies the reference is no longer needed. This parameter is always present

NOTE 1 – Service primitives of several different facilities have request references. In the GIF, the request references of each facility are considered distinct. In the mapping to a particular protocol, a single set of identifiers may be used. In such a case, the requirement for unambiguity covers all the relevant primitives.

NOTE 2 – In general, the request reference is globally unambiguous, but if the specific protocol establishes an association, any transmitted value for the request reference need only be unambiguous in the scope of that association. If only one primitive defined as having a request reference is outstanding on an association at a time, the request reference can be entirely implicit in the end-points of the association and no value is transmitted in the protocol.

11.2 Location-advise

11.2.1 Purpose

The location-advise primitives are used by a server-side to answer a location-query from a client-side.

11.2.2 Location-advise submit

The location-advise submit primitive occurs at a interworking interface of a server-side binder object.

11.2.3 Location-advise deliver

The location-advise deliver primitive occurs at a interworking interface of a binder object supporting a client object.

11.2.4 Parameters

The parameters of the location-advise primitive are:

request reference: the request reference of the location-query primitive to which this is the reply, and which identified the target object in its object_Id parameter.

location result: Shall have one of the following values:

- object unknown;
- object here;
- forward.

The value "object unknown" shall be used if this is a reply to a location-query primitive and the object identified in the location-query primitive is unknown to the server.

The value "object here" shall be used if this is a reply to a location-query primitive and the server can directly receive request primitives for the object identified in the location-query primitive.

The value "forward" shall be used if the location information parameter is present. This value can be qualified with information concerning how long the location information will remain valid or other attributes of the forwarding information.

NOTE 1 – The use of this qualifying information is not specified in this Recommendation | International Standard. It may result in further interactions to update stored interface references.

location information: This parameter shall be present if and only if the location result parameter is "forward". It shall contain an interface reference that can be used as the target for requests to the object identified in the location-query or request primitive to which this locate-advise primitive is the reply.

NOTE 2 – There is in general no guarantee that the location information gives correct information, or how long it will remain valid for. The location result parameter can be qualified to indicate the period and scope of validity.

NOTE 3 – There is no necessary syntactic relationship between any component fields of the original interface reference in the location-request and that returned in the location-advise.

11.2.5 Effects

Following the fault primitive, the request reference is no longer needed.

11.3 Location-cancel

11.3.1 Purpose

The location-cancel primitive is used by the sender of a location-query to inform the other side that it no longer requires a reply.

11.3.2 Location-cancel submit

The location-cancel submit primitive occurs at a binder object which has issued a location-query primitive and has not subsequently received a location-advise deliver with a matching request reference parameter.

The client side may receive a location-advise deliver with the matching request reference after issuing a location-cancel submit.

11.3.3 Location-cancel deliver

The location-cancel deliver primitive occurs at a binder object.

The location-cancel deliver primitive is advisory. The receiver can, but need not issue a location-advise submit with the same request reference after receiving a location-cancel deliver. The sender of the location-cancel may receive a location-advise deliver after issuing a location-cancel submit primitive.

11.3.4 Parameters

The parameters of the location-cancel primitive is:

request reference: the request reference of the location-query primitive that is being location-cancelled.

NOTE – The request reference does NOT become "no longer needed" after the location-cancel primitive is used.

11.4 Location-abort

11.4.1 Purpose

The location-abort primitive is a deliver-only primitive which signals to the binder object that the binder liaison has terminated. It can be received during attempted establishment of the binder liaison or after establishment is complete.

11.4.2 Location-abort deliver

The location-abort deliver primitive occurs at a client or server-side binder object.

11.4.3 Parameters

The parameter of the location-abort primitive is only present if the primitive is received during liaison establishment:

request reference: the request reference of the location-query primitive that was pending.

If the binder liaison is established (i.e. there is no outstanding location-query), there are no parameters on the location-abort-deliver primitive.

11.4.4 Effects

The binder liaison no longer exists.

The request reference, if present, become "no longer needed" after the location-abort deliver primitive.

11.5 State table for the Location Facility

The Location Facility is asymmetric between client and server-side. For each side, a separate state machine exists for each request reference. Primitives are assigned to the state machine that has the corresponding request reference.

11.5.1 Client-side state table

Table 4 – Client-side table for location facility

State / Incoming event	Idle	querying	cancelled
location-query submit	querying		
location-advise deliver		idle	idle
cancel submit		cancelled	
abort deliver	idle	idle	idle

11.5.2 Server-side state table

Table 5 – Server-side table for location facility

State / Incoming event	Idle	queried
location-query deliver	queried	
location-advise submit		idle
cancel deliver	idle	idle
abort deliver	idle	idle

12 Association management facility

This clause defines primitives that are concerned with the establishment and use of an association, rather than correspond to support of a particular computational interaction.

12.1 Association-request

12.1.1 Purpose

The association-request primitives are used to attempt to establish an association between two protocol objects.

12.1.2 Association-request submit

The association-request submit primitive occurs at the interworking interface of a protocol object that is attempting to establish an association. This is the initiating object for the association-request.

12.1.3 Association-request deliver

The association-request deliver primitive occurs at the interworking interface of a protocol object that is identified by the parameters of the corresponding association-request submit. This is the responding object for the association-request.

12.1.4 Parameters

The parameters of the association-request primitive are:

protocol object location: a representation of the address of the responder protocol object (or interceptor object). The exact form of the interface reference is protocol-dependent, and may be transformed by the protocol object. This transformation may involve interaction with other engineering objects.

request reference: an identification of this association-request that can be used by subsequent primitives to unambiguously refer to this association-request among all association-request primitives. The request reference shall remain unambiguous until the occurrence of an primitive that specifies the reference is no longer needed. This parameter is always present.

association reference: an identification of the association to be established that can be used by subsequent primitives to unambiguously refer to this association among all associations or requested associations. This parameter is always present.

NOTE – The relation between request reference and association reference is protocol-dependent. The request reference could be mapped to an incomplete association reference, with the association reference completed by the responder. Alternatively, two independent fields could be used.

12.2 Association-accept

12.2.1 Purpose

The association-accept primitives are used to successfully complete the establishment of an association between two protocol objects that was initiated by the association-request primitives.

12.2.2 Association-accept submit

The association-accept submit primitive occurs at the interworking interface of a protocol object which the responding object of an association-request deliver.

After the association-accept submit primitive has occurred, the association is established from the perspective of this protocol object that was responder to the association-request deliver.

12.2.3 Association-accept deliver

The association-accept deliver primitive occurs at the interworking interface of a protocol object which the initiating object of an association-request submit.

After the association-accept deliver primitive has occurred, the association is established from the perspective of this protocol object that was the initiating object of the association-request submit.

12.2.4 Parameters

The parameters of the association-accept primitives are:

request reference: this shall be the request reference from the association-request primitive to which it is the reply.

association reference: this shall be the association reference (or the completion o the association reference) from the association-request primitive to which it is the reply.

12.3 Association-reject

12.3.1 Purpose

The association-reject primitives ends an attempt to establish an association, and the association is not established.

12.3.2 Association-reject submit

The association-reject submit primitive occurs at the interworking interface of a protocol object which was the responding object of an association-request submit.

12.3.3 Association-reject deliver

The association-reject deliver primitive occurs at the interworking interface of a protocol object which the initiating object of an association-request submit.

An association-reject deliver can occur as a result of an association-reject submit at the protocol object that was responder to the corresponding association-request deliver. An association-reject deliver also occurs when the underlying communications are unable to support the establishment of the association.

12.3.4 Parameters

The parameter of the association-reject primitive are:

request reference: this shall be the request reference from the association-request primitive to which it is the reply.

12.4 Association-close

12.4.1 Purpose

The association-close primitives are used by a client-side protocol object that has an association with a server-side protocol object to inform the server-side that the association will no longer be used.

12.4.2 Association-close submit

The association-close submit primitive occurs at the interworking interface of a protocol object.

The issue of the association-close submit primitive has the effect of cancelling any request submit or locate-query submit primitives previously issued by the initialling object for which no matching result deliver or locate-advise deliver has been received.

Following the association-close submit primitive, the association is destroyed from the perspective of the initiating protocol object. No further submit primitives shall be issued and the occurrence of an event that would otherwise correspond to a deliver primitive shall be ignored.

12.4.3 Association-close deliver

The association-close deliver primitive occurs at the interworking interface of a protocol object.

Following the association-close deliver primitive, the association is destroyed from the perspective of the server-side protocol object. No further submit primitives shall be issued and the occurrence of an event that would otherwise correspond to a deliver primitive shall be ignored. Any request deliver or locate-query deliver primitive that has previously been received and not replied to is cancelled, no reply will be sent.

12.4.4 Parameter

The parameter of the association-close primitive is:

association reference: this shall be the association reference from the association-accept primitive which completed the establishment of the association.

12.5 Association-abort

12.5.1 Purpose

The association-abort submit and deliver primitive terminate the association. The association-abort deliver primitive can occur as a consequence of an association-abort submit issued by the other side, or can be is initiated by the underlying communication mechanisms. In this latter case, assuming a protocol object does not itself fail, the underlying communications will initiate association-abort deliver primitives to the protocol objects on both sides of the association.

12.5.2 Association-abort submit

The association-abort submit primitive occurs at the interworking interface of a client-side or server-side protocol object.

The primitive can occur on an association that is being established or on one that has been established.

Following the association-abort submit primitive, the association is destroyed from the perspective of the protocol object.

12.5.3 Association-abort deliver

The association-abort deliver primitive occurs at the interworking interface of a client-side or server-side protocol object. The primitive can occur as a consequence of an association-abort submit issued by the other side, or as a consequence of events in the underlying communications.

The primitive can occur on an association that is being established or on one that has been established.

Following the association-abort deliver primitive, the association is destroyed from the perspective of the protocol object.

12.5.4 Parameter

The parameters of the association-abort primitives depend on when the primitive occurs. If it occurs prior to the issue or receipt of an association-accept or association-reject primitive, the request reference shall be present. Otherwise the association reference shall be present:

request reference: this shall be the request reference from the preceding association-request primitive.

association reference: this shall be the association reference from the association-accept primitive which completed the establishment of the association.

12.6 State table for the Association Management Facility

The Association Management Facility is symmetric between client-side and server-side. Separate tables are shown for clarity.

A separate state machine exists for each association, distinguished by the association reference. Association references are distinct for each initiator and consequently "collisions" of association requests do not occur - they would be treated as the attempted establishment of two associations.

The "data transfer" event represents the transmission of protocol associated with any of the other facilities, where this is not piggy-backed on the association management primitive.

Table 6 – State table for association management facility

State / Incoming event	Idle	assn req pend	assn ind pend	established
association-request submit	assn req pend			
association-request deliver	assn ind pend			
association-accept submit			established	
association-accept deliver		established		
association-reject submit			idle	
association-reject deliver		idle		
association-close submit				idle
association-close deliver	idle			idle
data transfer submit				established
data transfer deliver				established
association-abort submit		idle	idle	idle
association-abort deliver		idle	idle	idle

Annex A

Mapping to CORBA GIOP and IIOP

(This annex forms an integral part of this Recommendation | International Standard)

A.1 Introduction

The General Inter-Orb Protocol (GIOP) is defined in [CORBA 2], which also defines the specific mapping of GIOP to TCP/IP in the Internet Inter-Orb Protocol (IIOP). This annex defines the mapping of the GIF to GIOP and IIOP.

The mapping defined in this annex is intended to be applicable to any version of GIOP and IIOP. Facilities present only in later versions of GIOP (e.g. fragmentation, marshalling optimization) that do not affect the semantic content and parameters of the messages are considered to be implicit. The different forms of Location-forward response, if supported by the GIOP version, qualify the parameters of the GIF primitives. The bi-directional GIOP facility (supported from GIOP 1.2) is included. The use or non-use of these features is dependent on the GIOP version negotiated on the connection.

NOTE – It is recognized that future versions of GIOP could add features that would require changes to GIF.

A.2 Conventions

The Message Body and Message Header of a particular GIOP message can be referred to as the <name of message> Body and <name of message> Header. e.g. "Reply Body" means the Message Body of the Reply message.

A.3 Generic Inter-Orb Protocol

GIOP is a connection-oriented protocol. The specification defines the format of a set of messages which are transmitted on an underlying association which is assumed to be reliable. GIOP messages do not themselves provide support for ensuring the reliable transmission of the messages. The encoded GIOP messages are self-delimiting - they do not require the underlying association to support discrete data units.

GIOP can support computational operations whose signature is defined in ODP IDL (see ITU-T Rec. X.920 | ISO/IEC 14750). GIOP includes encoding rules for the parameters of the computational operations.

A.3.1 Mapping of GIF primitives to GIOP messages

The service primitives of the Basic Interworking and Location facilities, and also the association-close primitives are mapped to the sending and receiving of GIOP messages on an established association. The submit primitives correspond to the sending of a GIOP message, the deliver primitives to the receipt of the GIOP message. The correspondence between these service primitives and GIOP messages is shown in Table A.1.

Table A.1 – Mapping of GIF primitives to GIOP messages

GIF primitive	GIOP message	Comments
request	Request	implicit location-query is piggy-backed; see text
result	Reply	
location-query	LocateRequest	
location-advise	LocateReply	when reply to explicit location-query
	Reply	if location-query was piggy-backed; see text
cancel	Cancel	
location-cancel	Cancel	if location-query is cancelled
association-close	CloseConnection	