INTERNATIONAL STANDARD

ISO/IEC 13235-1

First edition
1998-12-15

Information technology — Open Distributed Processing — Trading function: Specification

Technologies de l'information — Traitement distribué ouvert — Fonction commerciale: Spécifications

Reference number
ISO/IEC 13235-1:1998(E)

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.

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.

International Standard ISO/IEC 13235-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 33, *Distributed application services*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.950.

ISO/IEC 13235 consists of the following parts, under the general title *Information technology — Open Distributed Processing — Trading function*:

— *Part 1: Specification*

— *Part 2: (TBD)*

— *Part 3: Provision of trading function using OSI Directory service*

Annexes A to D form an integral part of this part of ISO/IEC 13235.

Introduction

The rapid growth of distributed processing has lead to a need for a coordinating framework for the standardization of Open Distributed Processing (ODP). The Reference Model of Open Distributed Processing (RM-ODP) provides such a framework. It defines an architecture within which support of distribution, interoperability and portability can be integrated.

One of the components of the architecture (described in RM-ODP Part 3: Architecture) (ITU-T Rec. X.903 | ISO/IEC 10746-3) is the ODP Trading function. The trading function provides the means to offer a service and the means to discover services that have been offered. This Recommendation | International Standard provides an architecture for systems implementing the trading function and the specification of interfaces within the architecture.

NOTE – The specification of computational interfaces in this Recommendation | International Standard is technically aligned with the OMG Trading Object Service.

The goals of this Recommendation | International Standard are:

- to provide a standard which is independent of any implementation;
- to ensure implementations are capable of being made to interoperate (i.e. can be federated);
- to provide sufficient detail to allow conformance claims to be assessed.

Annex A is a normative ODP-IDL specification of the trading function interface signatures.

Annex B is a normative specification of the ODP trading function constraint language.

Annex C is a normative specification of the ODP trading function constraint recipe language.

Annex D is an informative description of a Service Type Repository.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN DISTRIBUTED PROCESSING – TRADING FUNCTION: SPECIFICATION

1 Scope and field of application

The scope of this Recommendation | International Standard is:

- an enterprise specification for the trading function;
- an information specification for the trading function;
- a computational specification for traders (i.e. objects providing the trading function);
- conformance requirements in terms of conformance points.

It is not a goal of this Recommendation | International Standard to state how the trading function should be realized. Therefore this Recommendation | International Standard does not include an engineering specification.

The field of application for this Recommendation | Intenational Standard is any ODP system in which it is required to introduce and discover services incrementally, dynamically and openly.

2 Normative References

The following Recommendations and International Standards contain provisions which, trough 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.

- ITU-T Recommendation X.901 (1997) | ISO/IEC 10746-1:1998, *Information technology – Open distributed processing – Reference Model: Overview.*
- 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 14750:1998, *Information technology – Open Distributed Processing – Interface Definition Language.*
- ISO/IEC 13568[1)], *Information technology – The Z Specification Language.*

3 Notations

The information specification of the trading function is described using the Z formal description language. The signature of the computational interface for the trading function is described using ODP Interface Definition Language, in clause 8 and in Annex A.

4 Definitions

4.1 Definitions from ITU-T Rec. X.902 | ISO/IEC 10746-2

This Specification is based on the framework of abstractions and concepts developed in RM-ODP and makes use of the following definitions from RM-ODP Part 2: Foundations (see ITU-T Rec. X.902 | ISO/IEC 10746-2).

a) action;
b) activity;
c) behaviour;
d) behavioural compatibility;
e) binding;
f) client object;
g) conformance point;
h) contract;
i) domain;
j) establishing behaviour;
k) failure;
l) identifier;
m) initiating object;
n) instance;
o) interaction;
p) interface;
q) interface signature;
r) name;
s) object;
t) obligation;
u) ODP system;
v) permission;
w) policy;
x) prohibition;
y) quality of service;
z) reference point;
aa) responding object;
bb) role;
cc) server object;
dd) subtype;
ee) supertype;
ff) template;
gg) template type;
hh) trading;
ii) transparency;
jj) type;
kk) viewpoint.

4.2 Definitions from ITU-T X.903 | ISO/IEC 10746-3

This Specification is based on the framework of abstractions and concepts developed in RM-ODP and makes use of the following definitions from RM-ODP Part 3: Architecture (see ITU-T Rec. X.903 | ISO/IEC 10746-3).

a) community;

b) computational interface template;

c) computational viewpoint;

d) dynamic schema;

e) engineering viewpoint;

f) enterprise viewpoint;

g) exporter;

h) information viewpoint;

i) invariant schema;

j) schema;

k) service export;

l) service import;

m) service offer;

n) static schema;

o) technology viewpoint;

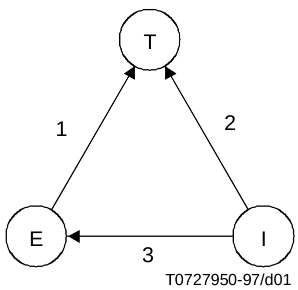
p) <X> federation.

5 Overview of the ODP Trading Function

In the context of the ODP goal of providing distribution transparent utilization of services over heterogeneous platforms and networks, the role of the Trading Function is to allow users to find potential services. It is a corollary of distribution that the finding of services will occur dynamically.

The ODP trading function facilitates the offering and the discovery of instances of interfaces which provide services of particular types. A trader is an object that supports the Trading Function in a distributed environment. It can be viewed as an object through which other objects can advertise their capabilities and match their needs against advertised capabilities. Advertising a capability or offering a service is called "export". Matching against needs or discovering services is called "import". Export and import facilitate dynamic discovery of and late binding to services.

To export, an object gives the trader a description of a service together with the location of an interface at which that service is available. To import, an object asks the trader for a service having certain characteristics. The trader checks against the descriptions of services and responds to the importer with the location(s) of matched service interface(s). The importer is then able to interact with a matched service. These interactions are shown in Figure 1.



Sequence of interactions:
1. Export
2. Import
3. Service Interaction

Figure 1 – Interaction between the trader and its clients

The service interaction could be decoupled from the trading interactions (export and import) by modelling a service provider object and a service user object explicitly. This would imply interactions between service provider and exporter and between importer and service user that are trading actions, as defined in ITU-T Rec. X.902 | ISO/IEC 10746-2. However, these implied interactions need not conform to this Specification.

Due to the sheer number of service offers that will be offered worldwide, and the differing requirements that users of the trading service will have, it is inevitable that the trading service will be split up and that the service offers will be partitioned.

Each partition will, in the first instance, meet the trading needs of a community of clients (exporters and importers). Where a client needs a scope for its trading activities that is wider than that provided by one partition, it will access other partitions either directly or indirectly. Directly means that the client interacts with the traders handling those partitions. Indirectly, means that the client interacts with one trader only and this trader interacts with other traders responsible for other partitions. The latter possibility is referred to as federation of traders. In some cases, interceptors may be required between federated traders.

A user of a trader that interoperates with other traders, may associate with only one trader, and can transparently access the service offers of other traders with which that trader can interoperate .

Thus, the trading function in an ODP environment allows:

- objects to export (advertise) services;
- objects to import information about one or more exported services, according to some criteria;
- federation of traders.

5.1 Diversity and scalability

The concept of trading to discover new services is one that is applicable in a wide range of scenarios. A trader may contain a large number of offers of service and its implementation may be inclined to be based upon a database. Or, a trader may contain a few offers only and so be implementable as a memory resident trader. These two cases exhibit different qualities: availability and integrity in the first case and performance in the second. The variation in these scenarios illustrates the need for scalability, both upwards for very big systems and downwards for small, fast systems.

To discover any arbitrary offer of service, a trader needs all offers to be, in some sense, visible to it. One partition cannot hold every offer, many will necessarily be held at other partitions, therefore, in addition to a number of offers, a trader must possess information about other partitions. However, there is no need for a trader to know about all other partitions. Some of this knowledge can be obtained indirectly via other traders.

The partitioning of the offer space and the limited knowledge held with one partition about other partitions is the basis for meeting requirements for both distribution and contextualisation of the trading function.

5.2 Linking traders

The requirements to contextualise the offer space and to distribute the trading function are both met by linking traders together. By linking to other traders, each trader makes the offer space of those traders implicitly available to its own clients.

Each trader has a horizon limited to those other traders to which it is explicitly linked. As those traders are in turn linked to yet more traders, a large number of traders are reachable from a given starting trader. The traders are linked to form a directed graph with the information describing the graph distributed among the traders. This graph is called the trading graph.

Links may cross domain boundaries: administrative, technological or whatever. Trading may thus be a federated system, i.e. one that spans many domains.

5.3 Policy

To meet the diverse requirements likely to be placed upon the trading function, some degrees of freedom are necessary when specifying the behaviour of a trader object. To accomplish this, and yet still meet the goals of this Specification, the concept of policy is used to provide a framework for describing the behaviour of any ODP conformant trading system.

This Specification identifies a number of policies and gives them semantics. Each policy partly determines the behaviour of a trader. When claiming conformance, an implementation may need to state which combination of policies will ensure conformant behaviour.

Policies may be communicated during interaction, in which case they relate to an expectation on subsequent behaviour.

6 Enterprise specification of the Trading Function

The scope of an enterprise specification is defined in RM-ODP Part 3: Architecture (see ITU-T Rec. X.903 | ISO/IEC 10746-3). This enterprise specification identifies the objectives and the policy statements that govern the activities of a trading function.

The objective of the trading function is to provide the means to offer and to discover instances of a particular type of service, with particular characteristics.

A trading community is comprised by members that have different roles, for example, trader, exporter and importer. An object can have several roles within the same community. For example, an object can both be an importer and an exporter.

The trading activities of the community are service exports and service imports. These activities are governed by a set of policies of the trading community. A service import activity may propagate from one trading community to another. In such a case the domains associated with these two traders are federated. These trader domain boundaries may coincide with other domain boundaries (e.g. type domain or security policy domain).

A policy is a set of rules with a particular objective. Each rule constrains some aspects of a trader•s behaviour consistent with the common objective. Members of the trading community are obliged to obey the rules of the policies. These rules provide the guidelines for decisions to satisfy the community's objectives. The rules are not prescribed in this Specification. The enterprise specification identifies the set of policies that limit the trader in certain type of behaviour. The policies identified provide a framework within which the trader object's behaviour may be implemented or configured.

6.1 Communities

6.1.1 **trading community**: A community of objects established for the purpose of trading and governed by a trading policy. The objects perform roles listed in 6.2.

A single trading community (at one level of abstraction) may be refined into a number of interworking trading communities at a second, more detailed level of abstraction. Subject to community policy, the interworking of trading communities at the detailed level is able to maintain the impression of the single abstract community, allowing objects with trader, importer or exporter roles in one subcommunity to interact with objects in any other subcommunity.

6.2 Roles

Objects may play the following roles within a trading community.

6.2.1 **trader**: A role which registers service offers from exporter objects and returns service offers upon request to importer objects according to some criteria.

6.2.2 **exporter**: A role which registers service offers with the trader object.

6.2.3 **importer**: A role which obtains service offers, satisfying some criteria, from the trader object.

6.2.4 **trader administrator**: A role which defines, manages, and enforces the trader policy of the trader object. The trader administrator is the controlling object of a trader domain (the trader and its set of service offers).

6.2.5 **service offer**: A role which maintains a description of a service.

NOTE – The description may be the basis of a future contract.

6.3 Activities

The following activities are relevant to a trading community.

6.3.1 **service export**: A chain of actions by an exporter object and the trader object which establish and terminate a liaison in which the trader object is permitted to provide the exporter object's service offer to a group of importer objects.

6.3.2 **service import**: A chain of actions between an importer object and the trader object in which the importer object obtains a number of service offers which meet some criteria.

6.4 Policies

The behaviours of enterprise objects within a trading community are governed by the policies of the trading community. Some policies govern trading activities, and some policies place constraints on other aspects of behaviour of a trader and other roles in the trading community, consistent with the common objective of the community. Where an activity involves interactions between objects, the resulting policy will be a compromise between the policies of the interacting objects. The compromise will be reached via a form of arbitration.

NOTE – For example, a trader object may be governed by policies such that it is obliged to propagate a search to a depth of 2 links, but it is also permitted to terminate a search after propagating a search to a depth of 1 link. If the trader object is permitted (or obliged) to meet an importer's requirement regarding depth of links to be traversed for a search, then there is a need for some rules to arbitrate between conflicting policies.

6.4.1 **Export activity policy**: A set of rules related to the service export activity (i.e. the offering of services so that they might subsequently be discovered by other objects).

The policy may include, amongst other things:

- an obligation for a service offer to be described in a specific way;
- a prohibition of specified service import activities from discovering the service offer;
- an obligation for a service offer providing rules to be evaluated as part of a service import activity.

Each exporter may have its own export policy. This would describe the exporter's expectation of a service export. Therefore, the service export activity is governed by both the trader's export policy and the exporter's export policy.

6.4.2 **Import activity policy**: A set of rules related to the service import activity (i.e. the attempt to discover offered services that meet a specified requirement.

The policy may include, amongst other things:

- an obligation to limit the use of resources, including duration of activity;
- permissions to propagate the service import to one or more interworking trading communities.

6.4.3 **Arbitration policy**: A set of rules to arbitrate on conflicting rules arising during trading activities.

The policy may include, amongst other things, an obligation to arbitrate in favour of the trader object's rules on:

- use of resources during service import;
- propagating service import activities.

6.4.4 **Service offer acceptance policy**: A set of rules restricting the set of service offers that will be accepted by the trader.

6.4.5 **Type management policy**: A set of rules related to the specification of types and the relationship between types.

NOTE 1 – The policy may be to defer to a type repository function with respect to either or both of these aspects.

NOTE 2 – Examples would be to use name equivalence or to use signature subtyping in type matching.

6.4.6 **Search policy**: A set of rules guiding the search for suitable service offers through the trading system.

6.5 Structuring rules

6.5.1 Community rules

In a trading community there must be an object which assumes a trader role (a trader object). Becoming a member of a trading community enables an object to interact with the trader object in an importer or an exporter role. An object may assume the exporter role, the importer role, or both the exporter and importer roles.

An "enterprise" may include multiple trading communities. An object can be a member of multiple trading communities. The trader object of one community may assume an importer or exporter role within another community of which it is a member.

The community may span several domains with respect to security, types, management, remuneration, etc.

Each trader, along with its set of service offers, is a trader domain. Thus, a set of trader domains which interoperate within a trading community is a federation of traders.

NOTE – Federated traders domains do not always require interceptors placed at their boundary in the engineering viewpoint.

6.5.2 Transfer rules

Exporter objects can export offers for services which they provide at their own interfaces, or may export offers for services provided by a distinct service provider object.

Importer objects can import service offers for their own use, or for use by distinct service user objects.

6.5.3 Delineation of authority rules

Each trader administrator object of a trading community has complete control over its own trader object.

The exporter object is responsible for the accuracy of its service offers.

For traders to be a member of an established federation of traders:

- one trader is not obliged to perform an activity initiated by another trader;
- each trader must have complete autonomy with respect to its own trader policies.

In particular, each trader determines its own trader search policies over the group of interworking traders.

6.5.4 Quality of service rules

The trader object is neither accountable nor responsible for the quality of services described in service offers.

A trader object may be obliged to ensure the timely removal of service offers.

NOTE – Two examples for achieving this are:

1) A trader service offer acceptance policy may oblige service offers to have an expiry date. The trader object is permitted to remove expired service offers.
2) A trader import policy may prohibit the trader object from returning service offers which have expired at the time of an import.

6.5.5 Matching rules

A service import requires computational interface signature type checking. It can, in addition, involve further levels of checking for subtype or supertype relationships, behavioural compatibility and environment constraints. Further checking on enterprise, information, engineering, and technology aspects may also be provided.

7 Information specification of the Trading Function

7.1 Overview

The scope of an information specification is defined in RM-ODP Part 3: Architecture (see ITU-T Rec. X.903 | ISO/IEC 10746-3). This information specification describes the types of information and the relationships between them which are required to define the ODP trading function. It uses the information language defined in RM-ODP and where appropriate interprets the language in terms of the formal specification notation Z. Paragraphs of formal notation are interspersed with English text in the usual Z style.

The information specification in this clause defines:

- basic concepts for information used in this Specification;
- static, invariant and dynamic schemata for this Specification.

7.2 Basic concepts

7.2.1 Interfaces

A service is offered at an interface. There is a need to for a service offer to identify the interface signature type and the interface identifier of the service interface.

7.2.1.1 Interface signature type

The interface signature type identifies the signature of interfaces of objects.

In Z, interface signature types are formally defined by introducing a given set to represent the values they can take:

[InterfaceSignatureType]

7.2.1.2 Interface identifier

An interface identifier identifies an interface at which a service is available or required.

In Z, interface identifiers are formally defined by a given set.

7.2.2 Service type

A service is a set of capabilities provided by an object at a computational interface. A service is an instance of a service type.

A service type definition consists of an interface signature type, a set of service property definitions, and a set of rules about the modes of the service properties.

Service property definitions are explicitly described in the formal specification in terms of names, value types and modes. The valid modes are:

- normal (read and write but optional presence);
- read-only (read but optional presence);
- mandatory (read and write mandatory presence); and
- read-only and mandatory (read only, mandatory presence).

A value type is a set of values.
[Name, Value]

$ValueType == \mathbb{P} Value$

$Mode ::= \{ normal, readonly, mandatory, readonly\_mandatory \}$

Service properties contain information about computational aspects (such as the behaviour and the environment of an interface) as well as describing the technology, engineering, information, and enterprise aspects of the service.

The formal definition of service type is given in the Z schema *ServiceType* . It groups together an interface signature type and a set of service property definitions.

ServiceType

$signature : InterfaceSignatureType$
$prop\_defs : Name \pfun (ValueType \times Mode)$

In Z, functions are used to extract the set of property names which must be present (i.e. they are *mandatory* or *readonly_ mandatory*) and the set of property names which cannot be modified (i.e. they are *readonly*, or *readonly_ mandatory*). These two functions are formally defined as follows:

mandatory_props *: ServiceType* → ℙ *Name*
readonly_props *: ServiceType* → ℙ *Name*

∀ *s: ServiceType* • ***mandatory_props*** *s* =
{ n : Name | second (s.prop_defs n) ∈ *{mandatory, readonly_mandatory}}*
∀ *s: ServiceType* • ***readonly_props*** *s* =
{n : Name | second (s.prop_defs n) ∈ *{readonly, readonly_mandatory}}*

7.2.3 Service type subtyping rules

The general rules for service subtyping for a conformant trader are as follows:

In the most general case, a service type **b** is a subtype of service type **a**, if, and only if:

- the interface signature type of **b** is a subtype of the interface signature type of **a**;
- all the named properties of **a** are in **b**;
- all of the named properties of **a** have a value type which is a supertype of the identically named property in **b**;
- all of the named properties of **a** have a mode which is a supertype of the mode of the identically named property in **b**.

NOTE – The above rules are equivalent to the normal ODP interface subtyping rules, if the properties are viewed as operations, with the type and mode as return arguments of the operations.

The Z representation requires that three relations are defined to represent interface signature subtyping, value supertyping and mode supertyping. The interface signature subtyping rules are those given in ITU-T

Rec. X.903 | ISO/IEC 10746-3 and are not further defined here. Formal definitions are given for supertyping relations across the modes and value types.

_ ***is_sig_subtype_of*** *_ : InterfaceSignatureType* ↔ *InterfaceSignatureType*
_ ***is_value_supertype_of*** *_ : ValueType* ↔ *ValueType*
_ ***is_mode_supertype_of*** *_ : Mode* ↔ *Mode*

∀ *a,b:Mode* • *a* ***is_mode_supertype_of*** *b* ⇔
(a,b) ∈ *{(normal,readonly),(normal,mandatory),(normal, readonly_mandatory),*
(readonly,readonly_mandatory), (mandatory,readonly_mandatory)}
∀ *a,b:ValueType* • *a* ***is_value_supertype_of*** *b* ⇔ *b* ⊆ *a*

__ ***is_subtype_of*** *__ : ServiceType*↔*ServiceType*

∀ *a,b : ServiceType* • *b* ***is_subtype_of*** ***a*** ⇔
b.signature ***is_sig_subtype_of*** *a.signature* ∧
dom a.prop_defs ⊆ *dom b.prop_defs* ∧
(∀ *n: dom a.prop_defs* •
first (a.prop_defs n) ***is_value_supertype_of*** *first (b.prop_defs n)* ∧
second (a.prop_defs n) ***is_mode_supertype_of*** *second (b.prop_defs n))*