AS/NZS ISO/IEC 13235.1:2006
ISO/IEC 13235-1:1998

AS/NZS ISO/IEC 13235.1:2006

Australian/New Zealand Standard™

# Information technology—Open Distributed Processing—Trading function

# Part 1: Specification

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Full document details: AS NZS ISO IEC 13235.1-2006

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT-015, Software and Systems Engineering. It was approved on behalf of the Council of Standards Australia on 27 October 2006 and on behalf of the Council of Standards New Zealand on 10 November 2006.
This Standard was published on 11 December 2006.

---

The following are represented on Committee IT-015:

Australian Computer Society
Australian Electrical and Electronic Manufacturers Association
Australian Society or Technical Communications
Australian Software Metrics Association
Engineers Australia/ACTS Joint Board in Software Engineering
Griffith University
National Association of Testing Authorities Australia
National ICT Australia
New Zealand Organisation for Quality
Software Quality Association, ACT
Software Quality Association, NSW
Systems Engineering Society of Australia
The University of Queensland
University of Auckland, NZ
University of South Australia
University of Technology, Sydney
Vendor Interests, NZ

---

## Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Web Shop at www.standards.com.au or Standards New Zealand web site at www.standards.co.nz and looking up the relevant Standard in the on-line catalogue.

Alternatively, both organizations publish an annual printed Catalogue with full details of all current Standards. For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia or Standards New Zealand at the address shown on the back cover.

---

*This Standard was issued in draft form for comment as DR 06504.*

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

Australian/New Zealand Standard™

# Information technology—Open Distributed Processing—Trading function

# Part 1: Specification

First published as AS/NZS ISO/IEC 13235.1:2006.

**COPYRIGHT**

© Standards Australia/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Jointly published by Standards Australia, GPO Box 476, Sydney, NSW 2001 and Standards New Zealand, Private Bag 2439, Wellington 6020

ISBN 0 7337 7904 2

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

# PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-015, Software and Systems Engineering.

The objective of this Standard is to provide an organization with a Reference Model of Open Distributed Processing RM-ODP) framework. It defines an architecture within which support of distribution, interoperability and portability can be integrated. The trading function provides the means to offer a service and the means to discover services that have been offered. This Standard provides architecture for systems implementing the trading function and the specification of interfaces within the architecture.

This Standard is identical with, and has been reproduced from ISO/IEC 13235-1:1998, *Information technology—Open Distributed Processing—Trading function: Specification*.

As this Standard is reproduced from an international standard, the following applies:

(a) Its number appears on the cover and title page while the international standard number appears only on the cover.

(b) In the source text 'this Recommendation/International Standard' should read 'this Australian/New Zealand Standard'.

(c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

| *Reference to International Standard* ISO/IEC | | *Australian/New Zealand Standard* AS/NZS | |
|---|---|---|---|
| 14750 | Information technology—Open Distributed Processing—Interface Definition Language | 14750 | Information technology—Open distributed processing—Interface definition language |

Only normative references that have been adopted as Australian or Australian/New Zealand Standards have been listed.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

# CONTENTS


| | | Page |
|---|---|---|
| 1 | Scope and field of application | 1 |
| 2 | Normative References | 1 |
| 3 | Notations | 1 |
| 4 | Definitions | 2 |
| 4.1 | Definitions from ITU-T Rec. X.902 \| ISO/IEC 10746-2 | 2 |
| 4.2 | Definitions from ITU-T X.903 \| ISO/IEC 10746-3 | 3 |
| 5 | Overview of the ODP Trading Function | 3 |
| 5.1 | Diversity and scalability | 4 |
| 5.2 | Linking traders | 4 |
| 5.3 | Policy | 4 |
| 6 | Enterprise specification of the Trading Function | 5 |
| 6.1 | Communities | 5 |
| 6.2 | Roles | 5 |
| 6.3 | Activities | 6 |
| 6.4 | Policies | 6 |
| 6.5 | Structuring rules | 6 |
| 7 | Information specification of the Trading Function | 7 |
| 7.1 | Overview | 7 |
| 7.2 | Basic concepts | 8 |
| 7.3 | Invariant schema | 12 |
| 7.4 | Static schema | 13 |
| 7.5 | Dynamic schemata | 13 |
| 8 | Computational specification of the Trading Function | 21 |
| 8.1 | Viewpoint correspondences | 22 |
| 8.2 | Concepts and data types | 22 |
| 8.3 | Exceptions | 35 |
| 8.4 | Abstract interfaces | 37 |
| 8.5 | Functional interfaces | 39 |
| 8.6 | Dynamic Property Evaluation interface | 55 |
| 8.7 | Trader object template | 56 |
| 9 | Conformance statements and reference points | 58 |
| 9.1 | Conformance requirement for trading function interfaces as server | 59 |
| 9.2 | Conformance requirements for query trader conformance class | 60 |
| 9.3 | Conformance requirements for simple trader conformance class | 60 |


Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

Page

9.4 Conformance requirements for stand-alone trader conformance class ..... 60
9.5 Conformance requirements for linked trader conformance class ..... 61
9.6 Conformance requirements for proxy trader conformance class ..... 61
9.7 Conformance requirements for full-service trader conformance class ..... 61
9.8 Conformance tests ..... 61
Annex A – ODP-IDL based specification of the Trading Function ..... 62
A.1 Introduction ..... 62
A.2 ODP Trading Function module ..... 62
A.3 Dynamic Property module ..... 69
Annex B – ODP Trading Function Constraint Language BNF ..... 71
B.1 Introduction ..... 71
B.2 Language basics ..... 71
B.3 The constraint language BNF ..... 72
Annex C – ODP Trading Function constraint recipe language ..... 75
C.1 Introduction ..... 75
C.2 The recipe syntax ..... 75
C.3 Example ..... 75
Annex D – Service type repository ..... 76
D.1 Introduction ..... 76
D.2 Service type repository ..... 76

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

# 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.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

NOTES

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

AUSTRALIAN/NEW ZEALAND STANDARD

# Information technology—Open Distributed Processing—Trading function

## Part 1: 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.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

# 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.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

### 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).

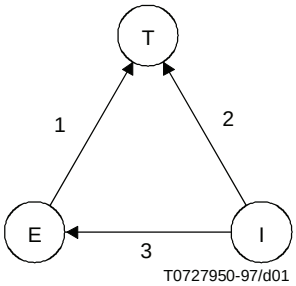
a) community;
b) computational interface template;
c) computational viewpoint;
d) dynamic schema;
e) engineering viewpoint;
f) enterprise viewpoint;
g) exporter;
h) information viewpoint;
i) invariant schema;
j) schema;
k) service export;
l) service import;
m) service offer;
n) static schema;
o) technology 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**

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

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.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

# 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.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

## 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.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

AVA Documents Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

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.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

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*

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

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

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:

$$
\begin{array}{l}
\textbf{\textit{mandatory_props}} : ServiceType \rightarrow \mathbb{P}\ Name \\
\textbf{\textit{readonly_props}} : ServiceType \rightarrow \mathbb{P}\ Name \\
\hline
\forall\ s: ServiceType \bullet \textbf{\textit{mandatory_props}}\ s = \\
\quad \{\ n : Name \mid second\ (s.prop_defs\ n) \in \{mandatory, readonly_mandatory\}\} \\
\forall\ s: ServiceType \bullet \textbf{\textit{readonly_props}}\ s = \\
\quad \{n : Name \mid second\ (s.prop_defs\ n) \in \{readonly, readonly_mandatory\}\}
\end{array}
$$

### 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.

$$
\begin{array}{l}
_\ \textbf{\textit{is_sig_subtype_of}}\ _ : InterfaceSignatureType \leftrightarrow InterfaceSignatureType \\
_\ \textbf{\textit{is_value_supertype_of}}\ _ : ValueType \leftrightarrow ValueType \\
_\ \textbf{\textit{is_mode_supertype_of}}\ _ : Mode \leftrightarrow Mode \\
\hline
\forall\ a,b:Mode \bullet a\ \textbf{\textit{is_mode_supertype_of}}\ b \Leftrightarrow \\
\quad (a,b) \in \{(normal,readonly),(normal,mandatory),(normal, readonly_mandatory), \\
\qquad\quad (readonly,readonly_mandatory), (mandatory,readonly_mandatory)\} \\
\forall\ a,b:ValueType \bullet a\ \textbf{\textit{is_value_supertype_of}}\ b \Leftrightarrow b \subseteq a
\end{array}
$$

$$
\begin{array}{l}
__\ \textbf{\textit{is_subtype_of}}\ __ : ServiceType \leftrightarrow ServiceType \\
\hline
\forall\ a,b : ServiceType \bullet b\ \textbf{\textit{is_subtype_of}}\ a \Leftrightarrow \\
\quad b.signature\ \textbf{\textit{is_sig_subtype_of}}\ a.signature\ \wedge \\
\quad dom\ a.prop_defs \subseteq dom\ b.prop_defs\ \wedge \\
\quad (\forall\ n: dom\ a.prop_defs \bullet \\
\qquad first\ (a.prop_defs\ n)\ \textbf{\textit{is_value_supertype_of}}\ first\ (b.prop_defs\ n) \wedge \\
\qquad second\ (a.prop_defs\ n)\ \textbf{\textit{is_mode_supertype_of}}\ second\ (b.prop_defs\ n))
\end{array}
$$

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

### 7.2.4 Service offer

A service offer advertises a service. It is an assertion made by an exporter about a service being offered for use by other objects at a computational interface. It consists of an instantiation of a service type, an identifier for the service offer and an identifier for the interface through which the service may be used. A service offer may also include a set of service offer property values.

For Z, a given set is introduced to represent the service offer identifiers that are unambiguous within a trading community.

*[ServiceOfferIdentifier]*

The formal Z definition of a service offer is given by the schema *ServiceOffer*. Each service offer must satisfy it's service type: i.e. the service offer must have a value for all the properties defined by the service type as mandatory or readonly-mandatory; and all those properties which are properties of the service type must have values drawn from the sets defined in the service type.

*ServiceOffer*

$$
\begin{array}{|l}
\hline
service_type : ServiceType \\
prop_vals : Name \pfun Value \\
interface_identifier : InterfaceIdentifier \\
service_offer_identifier : ServiceOfferIdentifier \\
\hline
\textbf{mandatory_props}\ service_type \subseteq dom\ prop_vals \\
\forall n: dom\ service_type.prop_defs \cap dom\ prop_vals \bullet \\
\quad prop_vals\ n \in first\ (service_type.prop_defs\ n) \\
\hline
\end{array}
$$

### 7.2.5 Criteria and constraints

Policies in the enterprise viewpoint are represented by criteria and constraints in the information viewpoint.

There are three aspects to specifying the set of service offers which are acceptable results of a search or select action. Two of these are filtering relations over the service properties and service offer properties. The first defines the essential properties which cannot be violated in order to match. The second is a preference which acts as a selection filter when there are many offers which match the essential properties. The third aspect is a scoping relation which restricts the set of service offers which are to be compared with the matching rules. These specifications may be provided by both the importer and the trading system, with the final specification being a combination of the two. The trading system is defined in 7.3.

#### 7.2.5.1 Matching criteria

Rules which are applied to the total set of service offers to yield a smaller set of acceptable service offers.

In Z , the matching criteria may be expressed as the set of all possible service offers which would satisfy the matching rules. The process of applying these rules to a set of service offers may then be represented by taking the intersection of two sets.

$MatchingCriteria == \mathbb{P} ServiceOffer$

#### 7.2.5.2 Preference criteria

Rules which are applied to the set of acceptable service offers to yield an ordered sequence of service offers.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

In Z, the preference criteria may be expressed as a total function from sets of service offers to a sequence of service offers. The application of these rules is formally described in 7.5.10.

*PreferenceCriteria* == ℙ*ServiceOffer* → **seq** *ServiceOffer*

#### 7.2.5.3 Scope criteria

Rules which restrict the set of service offers which are to be compared with the matching rules.

In Z, scope criteria may be expressed as a set of service offers.

*ScopeCriteria* == ℙ*ServiceOffer*

#### 7.2.5.4 Edge criteria

Rules which restrict the set of nodes reachable from a given node.

In Z, edge criteria can be expressed as associations:

*EdgeCriteria* == *Node x Node*

#### 7.2.5.5 Trader matching constraint

A constraint on the matching criteria imposed by policy of the trading system.

In Z, matching constraints can be specified in the same manner as matching criteria. Applying these constraints may be represented by set intersection.

#### 7.2.5.6 Trader preference constraint

A constraint on the preference criteria imposed by policy of the trading system.

In Z, preference constraints are represented by a total function between two sets of preference criteria. The application of this function is described in 7.5.10.

*PreferenceConstraint* == *PreferenceCriteria* → *PreferenceCriteria*

#### 7.2.5.7 Trader scope constraint

A constraint on the scope criteria imposed by trader policy.

In Z, scope constraints are represented by a set of service offers as for scope criteria.

#### 7.2.5.8 Trading system constraints

For the formal Z specification it is convenient to collect the constraints of the trading system together into the schema *TradingSystemConstraints*. This definition is used in 7.5.9.

*TradingSystemConstraints*

*trader_matching* : *MatchingCriteria*
*trader_scope* : *ScopeCriteria*
*trader_preference* : *PreferenceConstraint*

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

### 7.2.6 Search request

A search request is a specification of the importer policy applying to a particular search action.

In Z, a search request may be modelled as a schema consisting of the importer's matching, scope and preference criteria and the service type of the desired service. All offers which satisfy the importers matching requirements must have a service type which is a subtype of the specified service type. This definition is used in 7.5.9.

*Search Request*

*importer_matching* : *MatchingCriteria*
*importer_scope* : *ScopeCriteria*
*importer_preference* : *PreferenceCriteria*
*importer_service_type* : *ServiceType*

---

$\forall s{:}importer_matching \bullet s.service_type$ **is_subtype_of** *importer_service_type*

## 7.3 Invariant schema

The service offer space will be partitioned and associated with any one partition will be knowledge of a limited number of other partitions. This pattern is repeated for focusing on any given partition. In the information specification, this is modelled as a directed graph, where the nodes represent the partitions and the edges represent the knowledge relating partitions. This graph is referred to as the trading graph.

There may be any number of bases upon which to partition the offer space, over and above distribution. For example the partitioning may be based upon:

- properties of the location (e.g. machine architecture);
- properties of the service offers (e.g. a security classification);
- properties of the service (e.g. it's availability).

An edge may have properties that describe the perception of one partition when viewed from another partition.

The components of the trading system are:

- A set (*offers*) of service offers which are available for import;
- A set (*nodes*) of nodes into which these service offers are partitioned;
- A relationship (*edges*) between nodes to represent the edges of the trading graph, which governs the propagation of searches;
- Sets (*edge_properties*) of properties which are associated with the edges;
- Service offers are distinguished by their service offer identifiers, which are unique and unambiguous. This is captured by the invariants of the *TradingSystem* schema in Z.

Individual nodes are formally defined in Z by introducing a given set.

*[Node]*

All service offers must be allocated to one, and only one node. No overlapping or sub-set relations between the nodes are permitted. This restriction is captured by the partial function *partition* which maps each service offer in the trading system to a single node. An exporter may export the same service to more than one node, by creating another service offer with a different service offer identifier.

The computational viewpoint maps nodes into traders. This mapping is one-to-one (each node in the information viewpoint is a single trader object). Hence in the computational viewpoint an edge corresponds to a computational interface between traders.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

The following Z schema represents the state of the trading system.

*TradingSystem*

$$
\begin{array}{l}
offers : \mathbb{P}\ ServiceOffer \\
nodes : \mathbb{P}\ Node \\
partition : ServiceOffer \pfun Node \\
edges : Node \rightarrow Node \\
edge_properties : (Node \times Node) \pfun \mathbb{P}\ Property \\
\hline
dom\ partition = offers \\
ran\ partition \subseteq nodes \\
dom\ edges \cup \text{ran}\ edges \subseteq nodes \\
dom\ edge_properties = edges \\
\forall p,q : ServiceOffer \bullet p.service_offer_identifier = q.service_offer_identifier \Leftrightarrow p = q
\end{array}
$$

## 7.4 Static schema

Static schema applies the state of the trading system at a particular location in time.

For the consistency and completeness of the formal Z specification, the action to initialise a trading system is included.

When a trading system is created it has the null value for each state component:

*Initialize*

$$
\begin{array}{l}
TradingSystem' \\
\hline
offers' = \varnothing \\
nodes' = \varnothing \\
partition' = \varnothing \\
edges' = \varnothing \\
edge_properties' = \varnothing
\end{array}
$$

## 7.5 Dynamic schemata

This subclause presents dynamic information schemata which describe changes of state associated with the following actions:

- Export – Add a service offer to the service offer space of the trading system.
- Withdraw – Withdraw a service offer from the service offer space of the trading system.
- Modify Offer – Change the service property and service offer property values associated with a service offer whilst preserving the service offer identifier.
- Add Edge – Add an edge to the trading system's set of edges.
- Remove Edge – Remove an edge from the trading system's set of edges.
- Modify Edge – Change the properties of an edge.
- Add Node – Add a node to the trading system's set of nodes.
- Remove Node – Remove a node from the trading system's set of nodes.
- Search – Search for the subset of service offers which satisfy some matching criteria and scoping criteria.
- Select – A useful specialisation of search which returns a sequence of service offers ordered according to some preference constraint.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

### 7.5.1 Export

The dynamic schema Export describes the behaviour of adding a service offer to the trading system.

Successful export of a service offer is as follows. The new service offer is added to the existing set of service offers and is associated with a single node. The source of the service offer identifier component of the service offer is not identified: it is generated by the trader, rather than the exporter. The service offer identifier is passed back to the exporter. The edge relationships, and the properties associated with edges, are not changed.

The pre-condition for this action is that the new service offer identifier is not currently used in the trading system. Another pre-condition requires that the node with which the offer is to be associated exists in the trading system.

In Z, the *Export* schema represents the corresponding behaviour.

*ExportOK*

$$
\begin{array}{l}
\Delta TradingSystem \\
new_offer? : ServiceOffer \\
node? : Node \\
service_offer_identifier! : ServiceOfferIdentifier \\
\hline
\forall\, s : offers \bullet s.service_offer_identifier \neq new_offer?.service_offer_identifier \\
node? \in nodes \\
offers' = offers \cup \{new_offer?\} \\
partition' = partition \cup \{new_offer? \mapsto node?\} \\
service_offer_identifier! = new_offer?.service_offer_identifier \\
nodes' = nodes \\
edge_properties' = edge_properties \\
edges' = edges
\end{array}
$$

If the pre-conditions of the *ExportOK* schema are not met, the state of the trading system remains unchanged. In Z. this is captured in the following error schema, the pre-condition for which is the negation of the pre-condition of *ExportOK*.

*ExportError*

$$
\begin{array}{l}
\Xi TradingSystem \\
new_offer? : ServiceOffer \\
node? : Node \\
\hline
\exists\, s : offers \bullet s.service_offer_identifier = new_offer?.service_offer_identifier \lor \\
node? \notin nodes
\end{array}
$$

*Export* will succeed or fail depending on the pre-conditions of the above Z schemata.

$$Export \triangleq ExportOK \lor ExportError$$

### 7.5.2 Withdraw

The dynamic schema Withdraw removes a service offer from the trading system.

The offer is withdrawn from the set of offers. The edges and edge properties are unchanged. Note that this behaviour does not specify how the action was initiated. Thus the behaviour applies whenever an offer is withdrawn by the trader, the exporter or some other authorized agent.

The pre-condition for this action is that the service offer must exist in the trading system.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

In Z, the *Withdraw* schema represents the corresponding behaviour.

*WithdrawOfferOK*

$\Delta TradingSystem$
$old_offer? : ServiceOffer$

---

$old_offer? \in offers$
$offers' = offers \setminus \{old_offer?\}$
$partition' = \{old_offer?\} \lhd partition$
$nodes' = nodes$
$edges' = edges$
$edge_properties' = edge_properties$

If the pre-conditions of *WithdrawOfferOK* are not met, the state of the trading system remains unchanged.

*WithdrawOfferError*

$\Xi TradingSystem$
$old_offer? : ServiceOffer$

---

$old_offer? \notin offers$

*WithdrawOffer* will succeed or fail depending on the pre-conditions of the above Z schemata.

$WithdrawOffer \triangleq WithdrawOfferOK \lor WithdrawOfferError$

### 7.5.3 Modify Offer

The dynamic schema ModifyOffer defines the behaviour of modifying the service properties and service properties associated with a service offer. This behaviour is not simply Withdraw followed by Export, because it also guarantees that the service offer identifier is preserved.

A service offer may be modified in three ways:

- properties may be deleted, as long as they are not mandatory properties;
- new properties may be added, provided that they are not already assigned a value in the service offer;
- existing properties may be updated, provided that they are not readonly.

In Z, the *ModifyServiceOffer* schema represents the corresponding behaviour. The definition is given in several steps. First, a definition is given to modify a service offer by replacing the service property values, provided that the above constraints hold. The other components of the offer are not affected: in particular the service offer identifier is preserved. Note that this behaviour does not specify how the action was initiated. Thus the behaviour applies whenever an offer is modified by the trader, the exporter or some other authorized agent.

*ModifyServiceOfferOK*

$\Delta ServiceOffer$
$delete? : \mathbb{P}\ Name$
$new? : Name \pfun Value$
$update? : Name \pfun Value$

---

$delete? \cap \mathbf{mandatory_props}\ service_type = \varnothing$
$dom\ update? \cap \mathbf{readonly_props}\ service_type = \varnothing$
$delete? \cup dom\ update? \subseteq dom\ prop_vals$
$dom\ new? \cap dom\ prop_vals = \varnothing$
$prop_vals' = (delete? \lhd prop_vals) \oplus update? \cup new?$
$service_type'.signature = service_type.signature$
$interface_identifier' = interface_identifier$
$service_offer_identifier' = service_offer_identifier$

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

AVA Documents Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

Error conditions for this behaviour arise when an attempt is made to:

- update or modify a non-existant property;
- modify a property which is either read-only or mandatory read-only; or
- add a new property which already exists in the service properties.

If this happens, all properties of the service offer remain unchanged, and the interface signature is also unchanged.

*ModifyServiceOfferError*

$$
\begin{array}{|l}
\hline
\Xi ServiceOffer \\
delete? : \mathbb{P}\ Name \\
new? : Name \pfun Value \\
update? : Name \pfun Value \\
\hline
(\exists\, n : delete? \cup dom\ update? \bullet\ n \notin dom\ prop_vals) \\
\lor\ delete? \cap \textbf{\textit{mandatory_props}}\ service_type \neq \emptyset \\
\lor\ dom\ update? \cap \textbf{\textit{readonly_props}}\ service_type\ \neq \emptyset \\
\lor\ dom\ new? \cap dom\ prop_vals \neq \emptyset \\
\hline
\end{array}
$$

The Z technique of promotion is now employed to promote the *ModifyServiceOffer* schema so that it is applied to a specific offer in the trading system. The offer to be modified is identified by the definition of a framing schema, $\Phi$*ModifyOffer*. The framing schema states that:

- the identifier of the offer to be changed is known to the trading system;
- after the action, the selected offer has been changed to the new value. All other offers are unchanged and the offer remains in the original node;
- the edges and edge properties are unchanged.

$\Phi$*ModifyOffer*

$$
\begin{array}{|l}
\hline
\Delta TradingSystem \\
\Delta ServiceOffer \\
modified_offer? : ServiceOffer \\
\hline
modified_offer? \in offers \\
\theta ServiceOffer = modified_offer? \\
offers' = (offers \setminus \{\theta ServiceOffer\}) \cup \{\theta ServiceOffer'\} \\
partition' = \\
\quad (\{\theta ServiceOffer\} \lhd partition\ ) \cup \{\theta ServiceOffer' \mapsto partition\ \theta ServiceOffer\} \\
edges' = edges \\
edge_properties' = edge_properties \\
nodes' = nodes \\
\hline
\end{array}
$$

Error conditions for this behaviour arise when the pre-condition of the $\Phi$*ModifyOffer* does not hold. This is when an attempt is made to modify an offer which is not present in the trading system. The trading system remains in the same state.

*ModifyOfferError*

$$
\begin{array}{|l}
\hline
\Xi TradingSystem \\
modified_offer? : ServiceOffer \\
\hline
modified_offer? \notin offers \\
\hline
\end{array}
$$

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

Finally, the *ModifyOffer* behaviour is defined in terms of *ModifyServiceOffer*, the framing schema Φ*ModifyOffer* (which identifies a particular offer to be modified) and the error conditions defined in *ModifyOfferError*. The components of Δ *ServiceOffer* are hidden so that they do not appear in the declaration part of *ModifyOffer*, in line with the usual Z conventions.

$$ModifyOffer \triangleq ((ModifyServiceOfferOK \land \Phi ModifyOffer) \lor ModifyOfferError) \setminus (service_type, prop_vals, service_offer_identifier, interface_identifier, service_type', prop_vals', service_offer_identifier', interface_identifier')$$

### 7.5.4 Add Edge

The dynamic schema AddEdge defines the behaviour associated with adding an edge to the trading graph. An edge is added between the two nodes supplied, and the new edge properties are associated with this edge. The set of service offers and the nodes within the trading system remain unchanged.

The pre-conditions of this dynamic schema are that both nodes exist in the trading system, and that no edge already exists between these two nodes in the same direction.

In Z, the *AddEdge* schema represents the corresponding behaviour.

*AddEdgeOK*

$$\Delta TradingSystem$$
$$node1?, node2? : Node$$
$$new_edge_properties? : \mathbb{P} Property$$

---

$$\{node1?, node2?\} \subseteq nodes$$
$$(node1?, node2?) \notin edges$$
$$edges' = edges \cup \{node1? \mapsto node2?\}$$
$$edge_properties' = edge_properties \cup \{(node1?, node2?) \mapsto new_edge_properties?\}$$
$$offers' = offers$$
$$nodes' = nodes$$
$$partition' = partition$$

If the pre-conditions of *AddEdgeOK* are not met, the state of the trading system remains unchanged.

*AddEdgeError*

$$\Xi TradingSystem$$
$$node1?, node2? : Node$$

---

$$node1? \notin nodes \lor$$
$$node2? \notin nodes \lor$$
$$(node1?, node2?) \in edges$$

*AddEdge* will succeed or fail, depending on the pre-conditions of the above Z schemata.

$$AddEdge \triangleq AddEdgeOK \lor AddEdgeError$$

### 7.5.5 Remove Edge

The dynamic schema RemoveEdge removes an edge from the trading graph. The set of service offers and nodes within the trading system remain unchanged by this dynamic schema. The properties associated with the edge are removed from the *edge_properties*.

The pre-conditions for this action are that the edge supplied is in the current set of edges.

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

In Z, the *RemoveEdge* schema represents the corresponding behaviour.

*RemoveEdgeOK*

$$\Delta TradingSystem$$
$$old_edge? : Node \times Node$$

---

$$old_edge? \in edges$$
$$edges' = edges \setminus \{old_edge?\}$$
$$edge_properties' = \{old_edge?\} \lhd edge_properties$$
$$offers' = offers$$
$$nodes' = nodes$$
$$partition' = partition$$

If the pre-conditions of *RemoveEdgeOK* are not met, the state of the trading system remains unchanged.

*RemoveEdgeError*

$$\Xi TradingSystem$$
$$old_edge? : Node \times Node$$

---

$$old_edge? \notin edges$$

*RemoveEdge* will succeed or fail depending on the pre-conditions of the above Z schemata.

$$RemoveEdge \triangleq RemoveEdgeOK \lor RemoveEdgeError$$

### 7.5.6 Modify Edge

The dynamic schema ModifyEdge modifies the properties associated with an edge. The old properties associated with the edge are replaced by the new properties. The set of service offers and set of nodes remains unchanged, as do the properties associated with all other edges.

The pre-condition for this action is that the supplied edge must exist.

In Z, the *ModifyEdge* schema represents the corresponding behaviour.

*ModifyEdgeOK*

$$\Delta TradingSystem$$
$$edge? : Node \times Node$$
$$new_edge_properties? : \mathbb{P} Property$$

---

$$edge? \in edges$$
$$edge_properties' = edge_properties \oplus \{edge? \mapsto new_edge_properties?\}$$
$$edges' = edges$$
$$offers' = offers$$
$$nodes' = nodes$$
$$partition' = partition$$

If the pre-conditions of the *ModifyEdgeOK* are not met, the state of the trading system remains unchanged.

*ModifyEdgeError*

$$\Xi TradingSystem$$
$$edge? : Node \times Node$$

---

$$edge? \notin edges$$

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

*ModifyEdge* will succeed or fail depending on the pre-conditions of the above Z schemata.

$ModifyEdge \triangleq ModifyEdgeOK \lor ModifyEdgeError$

### 7.5.7 Add Node

The dynamic schema AddNode describes the behaviour of adding a node to the trading system.

The pre-condition of this action is that the node to be added must not already exist within the trading system. Since the new node was not an element of *nodes*, and hence not in the range of the *partition* function, no existing offers are mapped into the new node. This implies that new nodes do not contain offers.

In Z, the *AddNode* schema represents the corresponding behaviour.

*AddNodeOK*

$\Delta TradingSystem$
$new_node? : Node$

---

$new_node? \notin nodes$
$offers' = offers$
$nodes' = nodes \cup \{ new_node? \}$
$edges' = edges$
$edge_properties' = edge_properties$
$partition' = partition$

If the pre-conditions of *AddNodeOK* do not hold, the state of the trading system is unchanged.

*AddNodeError*

$\Xi TradingSystem$
$new_node? : Node$

---

$new_node? \in nodes$

*AddNode* will succeed or fail depending on the pre-conditions of the above Z schemata.

$AddNode \triangleq AddNodeOK \lor AddNodeError$

### 7.5.8 Remove Node

The dynamic schema RemoveNode defines the behaviour of removing a node from the trading system. The pre-conditions are that the node to be removed is present in the trading system, that it contains no service offers and that it no longer has edges to other nodes.

The computational behaviour may combine this dynamic schema with other dynamic schemata for removing service offers and edges.

In Z, the *RemoveNode* schema represents the corresponding behaviour.

*RemoveNodeOK*

$\Delta TradingSystem$
$old_node? : Node$

---

$old_node? \in nodes$
$old_node? \notin ran\ partition$
$old_node? \notin dom\ edges \cup ran\ edges$
$offers' = offers$
$nodes' = nodes \setminus \{old_node?\}$
$edge_properties' = edge_properties$
$edges' = edges$
$partition' = partition$

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006

If the pre-conditions of *RemoveNodeOK* do not hold, the state of the trading system is unchanged.

*RemoveNodeError*

$$\Xi TradingSystem$$
$$old_node? : Node$$

---

$$old_node? \notin nodes \lor$$
$$old_node? \in ran\ partition \lor$$
$$old_node? \in dom\ edges \cup ran\ edges$$

*RemoveNode* will succeed or fail depending on the pre-conditions of the above Z schemata.

$$RemoveNode \triangleq RemoveNodeOK \lor RemoveNodeError$$

### 7.5.9 Search

The dynamic schema Search is the behaviour which searches the trading system, restricted by some scoping criteria, and returns a set of service offers which satisfy some matching criteria. The action does not change the state of the trading system. Its output is a set of service offers. Its input is a search request.

Offers which satisfy the result of the search must satisfy the following conditions:

- they must have the correct service type or a subtype thereof;
- they must be contained in a node which is acceptable to both the requirements of the importer and the trading system's constraints, and is reachable from the starting point;
- they must match the importer's matching criteria;
- they must satisfy the trading system's matching constraints.

In Z, the *Search* schema represents the corresponding behaviour. The formal specification of Search represents the effect of matching criteria and search criteria as sets of service offers. The result of the search is obtained by simple set intersection. All offers which are returned must be reachable from the original starting point of the search. This property is expressed formally by stating that all such offers must be contained in a node which appears in the transitive closure of the *edges* relation.

*SearchOK*

$$\Xi TradingSystem$$
$$\Xi TradingSystemConstraints$$
$$SearchRequest?$$
$$starting_point? : Node$$
$$search_result! : \mathbb{P} ServiceOffer$$

---

$$starting_point? \in nodes$$
$$let\ e: EdgeCriteria \bullet e \subseteq edges$$
$$partition\ (\!|search_result!|\!) \subseteq$$
$$\{x : Node \mid (starting_point?, x) \in e^{+}\}$$
$$search_result! \subseteq$$
$$importer_matching? \cap trader_matching \cap importer_scope? \cap trader_scope$$

If an attempt is made to search starting from a node which does not exist, no service offers are returned and the state of the trading system is not changed.

*SearchError*

$$\Xi TradingSystem$$
$$\Xi TradingSystemConstraints$$
$$SearchRequest?$$
$$starting_point? : Node$$
$$search_result! : \mathbb{P} ServiceOffer$$

---

$$starting_point? \notin nodes$$
$$search_result! = \varnothing$$

Accessed by MONASH UNIVERSITY LIBRARY on 15 Jun 2015 (Document currency not guaranteed when printed)

Documents
Document Sharing Hub

Full document details: AS NZS ISO IEC 13235.1-2006