
**Information technology —
Programming languages, their
environments and system software
interfaces — Guidelines for the
preparation of language-independent
service specifications (LISS)**

*Technologies de l'information — Langages de programmation,
leurs environnements et interfaces du logiciel d'exploitation —
Lignes directrices pour l'élaboration de spécifications de service
indépendantes du langage (LISS)*

ISO/IEC TR 14369:2018

<https://standards.iteh.ai/catalog/standards/iso/c579454d-6771-44bf-a8ca-0d4c15b76f31/iso-iec-tr-14369-2018>



Reference number
ISO/IEC TR 14369:2018(E)

© ISO/IEC 2018

iTeh Standards
(<https://standards.itih.ai>)
Document Preview

ISO/IEC TR 14369:2018

<https://standards.itih.ai/catalog/standards/iso/c579454d-6771-44bf-a8ca-0d4c15b76f31/iso-iec-tr-14369-2018>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2018

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Published in Switzerland

Contents

Page

Foreword	vii
Introduction	viii
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 Abbreviated terms	4
5 Overview	4
5.1 Services, interfaces, service providers and service users	4
5.2 Information technology services	4
5.3 Services and language independence	5
5.4 Language-independent specifications	6
5.5 Problems of language dependence and inbuilt assumptions	7
5.5.1 General	7
5.5.2 Representational assumptions	7
5.5.3 Implementation assumptions	7
6 Guidelines on strategy	8
6.1 General	8
6.2 General guidelines	8
6.2.1 Guideline: Dependence of the interface on the service	8
6.2.2 Guideline: What to do when there are interoperability, concurrency, or time constraint issues	8
6.2.3 Guideline: Use of marshalling/unmarshalling	8
6.2.4 Guideline: Recruiting expertise from a variety of backgrounds	9
6.3 What to do if starting from scratch	9
6.3.1 General	9
6.3.2 Guideline: Avoidance of implementation assumptions	9
6.3.3 Specifying the service in language-independent form	9
6.3.4 Specifying the interface to the service in language-independent form	10
6.4 What to do if starting from an existing language-dependent specification	10
6.4.1 General	10
6.4.2 General guidelines	10
6.4.3 Converting an existing language-dependent specification of the service into language-independent form	12
6.4.4 Converting an existing implicit interface into an explicit language-independent interface	13
6.4.5 Specifying a language-independent interface to a service whose specification is language-dependent	14
7 Guidelines on document organization	15
7.1 General	15
7.2 Guideline: The general framework	15
7.2.1 General	15
7.2.2 Checklist of parts for inclusion	15
7.3 Guideline: Production and publication	16
7.4 Guideline: Document organization when starting from a language-specific specification	16
8 Guidelines on terminology	17
8.1 General	17
8.2 Guideline: The need for rigour	17
8.3 Guideline: The need for consistency	17
8.4 Guideline: Use of undefined terms	17
8.5 Guideline: Use of ISO 2382	17
8.6 Guideline: Use of definition by reference	18

8.7	Guideline: Terminology used in bindings	18
9	Guidelines on use of formal specification languages	18
9.1	Guideline: Use of a formal specification language	18
9.2	Checklist of formal specification languages	18
9.2.1	General	18
9.2.2	Estelle	18
9.2.3	Lotos	19
9.2.4	VDM-SL	19
9.2.5	Z	19
9.2.6	Extended BNF	20
9.3	Guideline: Using formal specifications from the outset	20
9.4	Guideline: Use of operational semantics	20
10	Guidelines on interoperability	21
10.1	General	21
10.1.1	Interoperability with what?	21
10.1.2	The nature of the interoperation	22
10.1.3	How interoperation is invoked	22
10.2	Guidelines on interoperability with other instantiations of the same service	22
10.2.1	Guideline: Identifying features affecting interoperability	22
10.2.2	Guideline: Precise definition and rigorous conformity requirements	22
10.2.3	Guideline: Importance of exchange values	23
10.3	Guidelines on interoperability with other services	23
10.3.1	General	23
10.3.2	Guideline: Interoperability with other services being defined at the same time	23
10.3.3	Guideline: Interoperability with a pre-defined service	23
11	Guidelines on concurrency issues	24
11.1	General	24
11.2	Guidelines on concurrency within the service specification	24
11.2.1	Guideline: Avoidance of unnecessary concurrency requirements	24
11.3	Guidelines on concurrency of interaction with service users	24
11.3.1	General	24
11.3.2	Guideline: Handling of concurrent service requests	25
11.3.3	Guideline: Number of concurrent service requests handled	25
11.3.4	Guideline: Order of processing of service requests	25
11.3.5	Guideline: Criteria for prioritizing service requests	25
11.4	Guidelines on concurrency requirements on bindings	25
11.4.1	General	25
11.4.2	Guideline: Avoidance of concurrency requirements	25
11.4.3	Guideline: Specification of unavoidable concurrency requirements	26
12	Guidelines on the selection and specification of datatypes	26
12.1	General	26
12.2	Guideline: Use of ISO/IEC 11404 General-Purpose Datatypes (GPD)	26
12.3	Guideline: Specification of datatype parameter values	26
12.4	Guideline: Treatment of values outside the set defined for the datatype	27
12.5	Guideline: Specification of operations on data values	27
12.6	Guideline: Recommended basic set of datatypes	27
12.7	Guideline: Specification of arithmetic datatypes	27
12.8	Guideline: Approach to language bindings of datatypes	28
12.9	Guideline: Avoidance of representational definitions	28
13	Guidelines on specification of procedure calls	28
13.1	General	28
13.2	Guideline: Avoidance of unnecessary operational assumptions or detail	29
13.3	Guideline: Use of ISO/IEC 13886 procedure calling model	29
13.4	Guidelines on the use of ISO/IEC 13886	29
13.4.1	General	29
13.4.2	Guideline: Selection of datatypes of parameters	30

13.4.3	Guideline: Selection of parameter passing modes.....	30
13.4.4	Guideline: Use of bindings to LIPC.....	31
13.5	Interfacing via remote procedure calling (RPC).....	31
13.5.1	General.....	31
13.5.2	Guideline: Avoid limiting the service specification because of constraints on the interface specification.....	31
13.5.3	Guideline: Specification of RPC interface.....	32
13.5.4	Guideline: Use of subsets.....	32
13.5.5	Guideline: Use of ISO/IEC 11578.....	32
13.6	Guideline: Guidance concerning procedure calling to those defining language bindings to the language-independent service specification.....	32
14	Guidelines on specification of fault handling.....	33
14.1	General.....	33
14.2	Guideline: Fault detection requirements.....	34
14.3	Checklist of potential faults.....	34
14.3.1	Invocation faults.....	34
14.3.2	Execution faults.....	34
14.4	Guideline: Recovery from non-fatal faults.....	35
15	Guidelines on options and implementation dependence.....	35
15.1	General.....	35
15.2	Guidelines on service options.....	36
15.2.1	Guideline: Optional service features.....	36
15.2.2	Guideline: Avoidance of assumptions about the use of the service.....	36
15.2.3	Guideline: Use of query mechanism.....	36
15.2.4	Guideline: Management of optional service features.....	36
15.2.5	Guideline: Definition of optional features.....	37
15.3	Guidelines on interface options.....	37
15.3.1	Guideline: Completeness of interface.....	37
15.3.2	Guideline: Interface to service with options.....	37
15.4	Guidelines on binding options.....	37
15.4.1	Guideline: Completeness of binding.....	37
15.4.2	Guideline: Binding to a service with options.....	37
15.4.3	Guideline: Binding to a language with optional features.....	38
15.5	Guidelines on implementation dependence.....	38
15.5.1	Guideline: Completeness of definition.....	38
15.5.2	Guideline: Provision of implementation options.....	38
15.5.3	Guideline: Implementation-defined limits.....	39
16	Guidelines on conformity requirements.....	40
16.1	General.....	40
16.2	Guidelines for specifying conformity of implementations of the service.....	41
16.2.1	Guideline: Avoidance of assumptions about the implementation language.....	41
16.2.2	Guideline: Avoidance of representational assumptions.....	41
16.2.3	Guideline: Avoidance of implementation model.....	41
16.2.4	Guideline: Requiring end results rather than methods.....	41
16.3	Guidelines for specifying conformity of implementations of the interface.....	41
16.3.1	Guideline: Requirements on implementation-defined aspects.....	41
16.4	Guidelines for specifying conformity of bindings.....	42
16.4.1	Guideline: Propagating requirements to conforming bindings.....	42
16.4.2	Guideline: Adherence to defined semantics.....	42
17	Guidelines on specifying a language binding to a language-independent interface specification.....	42
17.1	General.....	42
17.2	Guideline: Use of bindings to LID and LIPC.....	42
17.3	Guideline: Adherence to defined semantics.....	42
17.4	Guideline: Binding document organization.....	43
17.5	Guideline: "Reference card" binding documents.....	43

18	Guidelines on revisions	44
18.1	General	44
18.2	Kinds of change that a revision can introduce	44
18.2.1	General	44
18.2.2	Addition of a new feature	44
18.2.3	Change to the specification of a well-defined feature	44
18.2.4	Deletion of a well-defined feature	44
18.2.5	Deletion of ill-defined feature	44
18.2.6	Clarification of ill-defined feature	45
18.2.7	Change or deletion of obsolescent feature	45
18.2.8	Change of level definition	45
18.2.9	Change of specified limit to implementation-defined value	45
18.2.10	Change of other implementation requirement	45
18.2.11	Change of conformity clause	45
18.3	General guidelines applicable to revisions	45
18.3.1	Guideline: Revision compatibility	45
18.4	Guidelines on revision of the service specification	45
18.4.1	Guideline: Determining impact on interface and language bindings	45
18.4.2	Guideline: Minimising impact on interface and language bindings	46
18.4.3	Guideline: Use of incremental approach to revision	46
18.5	Guidelines on revision of the service interface	46
18.5.1	Guideline: Buffering unrevised bindings from changes	46
18.5.2	Guideline: Use of incremental amendments	46
18.6	Guidelines on revision of language bindings following revision of the service interface	46
18.6.1	Guideline: Buffering application programs from changes	46
18.6.2	Guideline: Use of incremental amendments	46
18.7	Guidelines on revision of a language binding following revision of the language	47
18.7.1	Guideline: Use of new language features	47
18.7.2	Guideline: Buffering “legacy” application programs from changes	47
18.7.3	Guideline: Buffering application programs by use of options	47
Annex A (informative)	Brief guide to language-independent standards	48
Annex B (informative)	Glossary of language-independent terms	51
Bibliography		64

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.

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

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

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

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This second edition cancels and replaces ISO/IEC TR 14369:1999 and ISO/IEC TR 14369:2014, of which it constitutes a minor revision.

The main changes compared to the previous edition are as follows:

- the reference section has been corrected/updated;
- editorial changes have been made to fully align with ISO/IEC Directives.

Introduction

This document is dedicated to Brian L. Meek in grateful recognition of his leadership and vision in the development of the concepts on programming language independent specifications, and his efforts in producing a set of documents in this area. Without his commitment this document never would have been published.

0.1 Background

This document provides guidance to those writing specifications of services, and of interfaces to services, in a language-independent way, in particular as standards. It can be regarded as complementary to ISO/IEC TR 10182, which provides guidance to those performing language bindings for such services and their interfaces.

NOTE 1 Here and throughout, “language”, on its own or in compounds like “language-independent”, means “programming language”, not “specification language” nor “natural (human) language”, unless explicitly stated.

NOTE 2 A “language-independent” service or interface specification can be expressed using either or both of a natural language like English or a formal specification language like VDM-SL or Z. In a sense, a specification can be regarded as “dependent” on VDM-SL, for example. The term “language-independent” does not imply otherwise, since it refers only to the situation where programming language(s) can otherwise be used in defining the service or interface.

The development of this document was prompted by the existence of an earlier draft IEEE Technical Report (IEEE TCOS-SCC Technical Report on Programming Language Independent Specification Methods, draft 4, May 1991). The TCOS draft was concerned with specifications of services in a POSIX systems environment, and as such contained much detailed POSIX-specific guidance; nevertheless it was clear that many of the principles, if not the detail, were applicable much more generally. This document was conceived as a means of providing such more general guidance. Because of the very different formats, and the POSIX-related detail in the TCOS draft, there is almost no direct correspondence between the two documents, except in the discussion of the benefits of a language-independent principles below. However, the spirit and principles of the TCOS draft were of great value in developing this document, and reappear herein, albeit in much altered and more general form.

NOTE 3 The TCOS draft has not in fact been published, as the result of an IEEE decision to concentrate activities in other POSIX areas.

0.2 Principles

Service or interface specifications that are independent of any particular language, particularly when embodied in recognized standards, are increasingly seen as an important factor in promoting interoperation and substitution of system components, and reducing dependence on and consequent limitations due to particular language platforms.

NOTE It is possible for a specification to be “independent” of a particular language in a formal sense, but still be dependent on it through inbuilt assumptions derived from that language which do not necessarily hold for other languages. The term “language-independent” here is meant in a much stronger sense than that, though complete independence from all inbuilt assumptions can be difficult if not impossible to achieve.

Potential benefits from language-independent service or interface specifications include:

- A language-independent interface specification specifies those requirements that are common to all language bindings to that interface, and hence provides a specification to which language bindings can conform.
- A language-independent interface specification is a re-usable component for constructing language bindings.
- A language-independent interface specification aids the construction of language bindings by providing a common reference to which all bindings can relate. Through this common reference it is possible to make use of pre-existing language bindings to language-independent standards

for common features such as datatypes and procedure calls, and to other language-independent specifications with related concepts.

- A language-independent service or interface specification provides an abstract specification of a service in isolation from language-dependent extensions or restrictions, and hence facilitates more rigorous modelling of services and interfaces.
- Language-independent service specifications facilitate the specification of relationships between one service and another, by making it easier to relate common concepts than is generally possible when the specifications are dependent on different languages.
- A language-independent interface specification facilitates the definition of relationships between different language bindings to a common service (such as requirements for interoperability between applications based on different languages that are sharing a common service implementation), by providing a common reference specification to which all the languages can relate.
- A language-independent interface specification facilitates the definition of relations between bindings to multiple services, including the requirements on management of multiple name spaces.
- A language-independent service or interface specification brings economic benefits by reducing the effort and resources needed to ensure compatibility and consistency of behaviour between implementations of the same service in different languages or between applications based on different languages using the same interface.

iTeh Standards (<https://standards.itih.ai>) Document Preview

[ISO/IEC TR 14369:2018](https://standards.itih.ai/catalog/standards/iso/c579454d-6771-44bf-a8ca-0d4c15b76f31/iso-iec-tr-14369-2018)

<https://standards.itih.ai/catalog/standards/iso/c579454d-6771-44bf-a8ca-0d4c15b76f31/iso-iec-tr-14369-2018>

Information technology — Programming languages, their environments and system software interfaces — Guidelines for the preparation of language-independent service specifications (LISS)

1 Scope

This document provides guidelines to those concerned with developing specifications of information technology services and their interfaces intended for use by clients of the services, in particular by external applications that do not necessarily all share the environment and assumptions of one particular programming language. The guidelines do not directly or fully cover all aspects of service or interface specifications, but they do cover those aspects required to achieve language independence, i.e. required to make a specification neutral with respect to the language environment from which the service is invoked. The guidelines are primarily concerned with the interface between the service and the external applications making use of the service, including the special case where the service itself is already specified in a language-dependent way but needs to be invoked from environments of other languages. Language bindings, already addressed by ISO/IEC TR 10182, are dealt with by providing advice on how to use the two documents together.

This document provides technical guidelines, rather than organizational or administrative guidelines for the management of the development process, though in some cases the technical guidelines can have organizational or administrative implications.

2 Normative references

There are no normative references in this document.

<https://standards.iteh.ai/catalog/standards/iso/c579454d-6771-44bf-a8ca-0d4c15b76f31/iso-iec-tr-14369-2018>

3 Terms and definitions

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

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

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <http://www.electropedia.org/>

3.1 client

application (typically a program in some language) which makes use of a *service* (3.13)

Note 1 to entry: The term “service user” (3.15) is often used in a similar sense, where “client” more often implies the physical computer system on which the application is running, rather than just the application itself.

3.2 datatype

set of values, usually accompanied by a set of operations on those values

3.3 formal language

formal *specification language* (3.16)

**3.4
interface**

mechanism by which a *service user* (3.15) invokes and makes use of a *service* (3.13)

**3.5
language**

programming language, not *specification language* (3.16) or natural (human) language, unless otherwise qualified

**3.6
language binding**

specification of the standard interface to a *service* (3.13), or set of services, for applications written in a particular programming language

**3.7
language-dependent**

state of making use of the concepts, features or assumptions of a particular programming language

**3.8
language-independent**

state of not making use of the concepts, features or assumptions of any particular programming language or style of language

**3.9
language processor**

entire computing system which enables a programming language user to translate and execute programs written in the language, in general consisting both of hardware and of the relevant associated software

[SOURCE: ISO/IEC TR 10176:2003, 3.1, modified — the words “denotes the” at the beginning of the definition, and the two Notes to entry were deleted.]

**3.10
mapping**

defined association between elements (such as concepts, features or facilities) of one entity (such as a programming language, or a specification, or a standard) with corresponding elements of another entity

Note 1 to entry: Mapping when used as a verb is a process of determining or utilizing a mapping.

Note 2 to entry: Mappings are usually defined as being from one entity into another. A language binding of a language L into a standard S usually incorporates both a mapping from L into S and a mapping from S into L.

Note 3 to entry: Depending on what is being mapped, a mapping is not necessarily one-to-one. This means that mapping an element E from system A into an element E' of system B, followed by mapping E' back into system A, does not necessarily get back to the original E. In such situations, if a two-way correspondence is to be preserved, the execution of the mappings needs to include recording the place of origin and returning to it.

**3.11
marshalling**

process of collecting the actual parameters used in a procedure call, converting them if necessary, and assembling them for transfer to the called *procedure* (3.12)

Note 1 to entry: This process is also carried out by the called procedure when preparing to return the results of the call to the caller.

Note 2 to entry: Marshalling can be regarded as being performed by a service user when preparing input values for a service provider, and by a service provider when preparing results for a service user, the service concerned being regarded as the procedure being called.

3.12**procedure**

subprogram which can return a value

Note 1 to entry: A procedure that returns a value is sometimes called a subroutine, a procedure that does not return a value is sometimes called a function.

Note 2 to entry: Some programming languages use different terminology.

3.13**service**

facility, or set of facilities, made available to *service users* (3.15) through an *interface* (3.4)

3.14**service provider****server**

computer system, or set of computer systems, that implements a *service* (3.13) and makes it available to *service users* (3.15)

Note 1 to entry: In this definition, “computer system” means a logical system, not a physical system; it can correspond to part of all of one or more physical computer systems.

Note 2 to entry: The term “server” is often used in a similar sense, though sometimes implying a physical computer system that has no other function than to provide its service

3.15**service user**

application (typically a program in some language) which makes use of a *service* (3.13)

Note 1 to entry: The term “client” (3.1) is often used in a similar sense, though sometimes implying the physical computer system on which the application is running, rather than just the application itself.

3.16**specification language**

formal language for defining the semantics of a service or an interface precisely and without ambiguity

3.17**unmarshalling**

process of receiving and disassembling transferred parameters, and converting them if necessary, to prepare the values for further use

Note 1 to entry: This process is carried out by the called procedure on receipt of the actual parameters for the call, and by the caller on receipt of the returned results of the call.

Note 2 to entry: Unmarshalling can be regarded as being performed by a service provider when receiving input values from a service user, and by a service user when receiving results from a service provider, the service concerned being regarded as the procedure being called.

3.18**Z**

<mathematics>complex numbers

Note 1 to entry: See ISO/IEC 10967-1.

3.19**Z**

type of formal *specification language* (3.16)

Note 1 to entry: It is pronounced “zed”.

Note 2 to entry: See ISO/IEC 13568.

4 Abbreviated terms

GPD	general-purpose datatypes, as defined in ISO/IEC 11404:2007
LID	language-independent datatypes, as defined in ISO/IEC 11404:1996; the LID specifications used in this document are identical to the corresponding specifications in ISO/IEC 11404:2007 (GPD)
LIPC	language-independent procedure calling, as defined in ISO/IEC 13886
RPC	remote procedure call, as defined in ISO/IEC 11578:1996

5 Overview

5.1 Services, interfaces, service providers and service users

The concept of a “service” is a very general one. In some contexts it is customary to use it in a restricted sense, e.g. when talking about “service industries” as contrasted with “manufacturing industries”. Despite such usages, almost any activity or behaviour can be regarded as a “service”, if it serves some useful purpose to do so (for example, manufacturing spoons can be regarded as a service for those needing spoons).

With the concept of a service come the concepts of a “service provider” and a “service user”. The provider performs the activity that constitutes the service; the user is the customer or the client for the service, for whom the service is performed. In the information technology field, the “client-server model” incorporates these concepts: the server provides, the client uses.

Between the service provider and the service user is an interface that allows them to communicate. The service user communicates through the interface the requirement for the service, and any relevant information (e.g. not only the need for spoons, but the number and size of spoons required), and the service provider communicates through the interface the response to the order for the service, and any additional information or queries (e.g. the spoons can be delivered in six days, do you want silver spoons or plastic spoons?). In the information technology field, such interfaces are usually explicit, realized in hardware or software or both. In the world in general, they are sometimes explicit, but sometimes subsumed in more general human or other interactions.

This distinction between provider and user (client and server) should not be assumed to correspond to identifiable distinct entities. The distinction, and the service interface, can be purely notional, and possibly not normally thought of in that way. The service itself can similarly not correspond to a distinct, separate activity, and again possibly not normally thought of as such; it can be subsumed in some other activity or group of activities, and can possibly be implicit.

Hence, for example, in a transaction between two parties, each one can be providing a service for the other: each is a client, and each a server. In another context, the provider is providing the service to itself; the provider is also the user. Though it is possible to subdivide the provider/user into a provider part and a user part when considering provision of the service, this can be inconvenient in other respects.

In summary, “client” and “server”, are roles that are carried out, rather than elements that necessarily need to be implemented separately. Though the term “client-server” is sometimes used in the information technology field in ways that are more specific than it is used here, it is important not to carry over assumptions from particular client-server models when reading this document. It is even more important not to assume that implementation of any service, in the sense used here, needs to be done using a client-server model.

5.2 Information technology services

The history of information technology has many instances of the technology, or a product, being used for very different purposes and in very different ways from those originally envisaged. The kinds of