INTERNATIONAL STANDARD

ISO/IEC 13886

First edition
1996-03-15

Information technology — Language-Independent Procedure Calling (LIPC)

Technologies de l'information — Appel de procédure indépendant du langage (LIPC)

Reference number
ISO/IEC 13886:1996(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organizations to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

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

International Standard ISO/IEC 13886 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces.*

Annexes A to D of this International Standard are for information only.

Introduction

The purpose of this International Standard is to provide a common model for language standards for the concept of procedure calling. It is an enabling standard to aid in the development of language-independent tools and services, common procedure libraries and mixed language programming. In mixed language applications, server procedures would execute on language processors operating in server mode, and the procedures would be called from language processors operating in client mode. Note that the languages need not be different, and if the processors are the same the model collapses into conventional single processor programming.

Most programming languages include the concepts of procedures and their invocation. The main variance between the methods used in various programming languages lies in the ways parameters are passed between the client and server procedures. Procedure calling is a simple concept at the functional level, but the interaction of procedure calling with datatyping and program structure along with the many variations on procedure calling and restrictions on calling that are applied by various programming languages transforms the seemingly simple concept of procedure calling into a more complex feature of programming languages.

The need for a standard model for procedure calling is evident from the multitude of variants of procedure calling in the standardized languages. The existence of this International Standard for Language-Independent Procedure Calling (LIPC) does not require that all programming languages should adopt this model as their sole means of procedure calling. The nominal requirement is for programming languages to provide a mapping to LIPC from their native procedure calling mechanism, and to be able to accept calls from other programming languages who have defined a mapping to this International Standard.

This International Standard is a specification of a common model for procedure calling. It is not intended to be a specification of how an implementation of the LIPC is to be provided. Also, it is important to note that it does not address the question of how the procedure call initiated by the client mode processor is communicated to the server mode processor, or how the results are returned. The model defined in this International Standard is intended for use by languages so that they may provide standard mappings from their native procedure model. This International Standard depends on the International Standard for Language-Independent Datatypes, ISO/IEC 11404, for the definition of the datatypes that are to be supported in the model for LIPC that it provides.

This page intentionally left blank

Information technology – Language-Independent Procedure Calling (LIPC)

1 Scope

This International Standard specifies a model for procedure calls, and a reference syntax for mapping to and from the model. This syntax is referred to as the Interface Definition Notation. The model defined in this International Standard includes such features as procedure invocation, parameter passing, completion status, and environmental issues relating to non-local references and state.

This International Standard does not specify:

- the method by which the procedure call initiated by the client mode processor is communicated to the server mode language processor;
- the minimum requirements of a data processing system that is capable of supporting an implementation of a language processor to support LIPC;
- the mechanism by which programs written to support LIPC are transformed for use by a data processing system;
- the representation of a parameter.

NOTE – Originally it was the intention to align the definitions and concepts of this International Standard with those of the RPC standard (ISO/IEC 11578). Unfortunately, in a late stage of the development process of the RPC standard it was decided to use for that standard a completely different approach. Hence the intended alignment did not materialize.

Annex D gives an overview of the differences between the concepts as defined by this International Standard and the RPC standard.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of current valid International Standards.

ISO 2375:1985, *Data processing – Procedure for registration of escape sequences.*

ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*

ISO/IEC 11404:1996, *Information technology – Programming languages, their environments and system software interfaces – Language-independent datatypes.*

ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

ISO/IEC 8825-1:1995, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

3 Definitions and Abbreviations

3.1 Definitions

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

3.1.1 actual parameter: A value that is bound to a formal parameter during the execution of a procedure.

3.1.2 association: Any mapping from a set of symbols to values.

3.1.3 box: A model of a variable or container that holds a value of a particular type.

3.1.4 client interface binding: The possession by the client procedure of an interface reference.

3.1.5 client procedure: A sequence of instructions which invokes another procedure.

3.1.6 complete procedure closure: A procedure closure, all of whose global symbols are mapped.

3.1.7 configuration: Host and target computers, any operating system(s) and software used to operate a processor.

3.1.8 execution sequence: A succession of global states s_1, s_2, ... where each state beyond the first is derived from the preceding one by a single create operation or a single write operation.

3.1.9 formal parameter: The name symbol of a parameter used in the definition of a procedure to which a value will be bound during execution.

3.1.10 global state: The set of all existing boxes and their currently assigned values.

3.1.11 global symbol: Symbol used to refer to values that are permanently associated with a procedure.

3.1.12 implementation defined: An implementation defined feature is a feature that is left implementation dependent by this International Standard, but any implementation claiming conformity to this International Standard shall explicitly specify how this feature is provided.

3.1.13 implementation dependent: An implementation dependent feature is a feature which shall be provided by an implementation claiming conformity to this International Standard, but the implementation need not to specify how the feature is provided.

3.1.14 input parameter: A formal parameter with an attribute indicating that the corresponding actual parameter is to be made available to the server procedure on entry from the client procedure.

3.1.15 input/output parameter: A formal parameter with an attribute indicating that the corresponding actual parameters are made available to the server procedure on entry from the client procedure and to the client procedure on return from the server procedure.

3.1.16 interface closure: A collection of names and a collection of procedure closures, with a mapping between them.

3.1.17 interface execution context: The union of the procedure execution contexts for a given interface closure.

3.1.18 interface reference: An identifier that denotes a particular interface instance.

3.1.19 interface type: A collection of names and a collection of procedure types, with a mapping between them.

3.1.20 interface type identifier: An identifier that denotes an interface type.

3.1.21 invocation association: The invocation association of a procedure closure <Image, Association> applied to a set of actual parameter values is the association of the closure augmented by a mapping of all local symbols to values and all formal parameter symbols to the corresponding actual parameter values. Thus it is a binding to values of all symbols in the procedure image for the duration of the invocation.

3.1.22 invocation context: For a particular procedure call, the instance of the objects referenced by the procedure, where the lifetime of the objects is bounded by the lifetime of the call.

3.1.23 marshalling: A process of collecting actual parameters, possibly converting them, and assembling them for transfer.

3.1.24 output parameter: A formal parameter with an attribute indicating that the corresponding actual parameter is to be made available to the client procedure on return from the server procedure.

3.1.25 parameter: A parameter is used to communicate a value from a client to a server procedure. The value supplied by the client is the actual parameter, the formal parameter is used to identify the received value in the server procedure.

3.1.26 partial procedure closure: A procedure closure, some of whose global symbols are not mapped. Procedure closures may be complete, with all global symbols mapped, or partial with one or more global symbols not mapped.

3.1.27 procedure: The procedure value.

3.1.28 procedure call: The act of invoking a procedure.

3.1.29 procedure closure: A pair <procedure image, association> where the association defines the mapping for the image's global symbols and no others.

NOTE – Procedure closures are the values of procedure type referred to in ISO/IEC 11404 - Language-Independent Datatypes.

3.1.30 procedure execution context: For a particular procedure, an instance of the objects satisfying the external references necessary to allow the procedure to operate, where these objects have a lifetime longer than a single call of that procedure.

3.1.31 procedure image: A representation of a value of a particular procedure type, which embodies a particular sequence of instructions to be performed when the procedure is called.

3.1.32 procedure invocation: The object which represents the triple: procedure image, execution context, and invocation context.

3.1.33 procedure name: The name of a procedure within an interface type definition.

3.1.34 procedure return: The act of return from the server procedure with a specific termination.

3.1.35 procedure type: The family of datatypes each of whose members is a collection of operations on values of other datatypes. Note, this is a different definition from procedure value.

3.1.36 procedure value: A closed sequence of instructions that is entered from, and returns control to, an external source.

3.1.37 processor: A compiler or interpreter working in combination with a configuration.

3.1.38 server procedure: The procedure which is invoked by a procedure call.

3.1.39 symbol: A program entity used to refer to a value.

3.1.40 termination: A predefined status related to the completion of a procedure call.

3.1.41 unmarshalling: The process of disassembling the transferred parameters, possibly converting them, for use by the server procedure on invocation or by the client procedure upon procedure return.

3.1.42 value: The set Value contains all the values that might arise in a program execution.

3.2 Abbreviations

3.2.1 ASN.1: Abstract Syntax Notation - One

3.2.2 IDN: Interface Definition Notation

3.2.3 LID: Language-Independent Datatypes, as defined in ISO/IEC 11404:1995.

3.2.4 LIPC: Language-Independent Procedure Calling

4 Conformance

A language processor may conform to this International Standard by mapping its native procedure calling mechanism to the LIPC model that this International Standard defines.

NOTE – The term "language processor" used in this clause may be extended to include anything which processes information and contains a procedure calling mechanism.

4.1 Modes of conformance

A language processor claiming conformance to this International Standard shall conform in either or both of the following ways.

4.1.1 Client mode conformance

In order to conform in client mode, a language processor shall allow programs written in its language to call procedures written in another language and supported by another processor, using the language-independent procedure calling (LIPC) as provided by clauses 5, 6 and 7 of this International Standard. In this case it is said to conform in (and be able of operating in) client mode. As part of this, the language processor shall define a mapping from its own procedure calling model to the LIPC model.

NOTE – If a program using the LIPC facility is to be portable between processors which conform in client mode, the program and processors will also need to conform to the relevant language standard and the relevant standards binding for that language to the LIPC and LID standards.

4.1.2 Server mode conformance

In order to conform in server mode, a language processor shall allow programs written in another language to call procedures written in its language (i.e. it will accept and execute procedure calls generated by another processor which is executing in a program that is written in that other language and which is operating in client mode, and return control to that client processor upon completion), using the language-independent procedure calling (LIPC) as provided by clauses 5, 6 and 7 of this International Standard. In this case it is said to conform in (and be able of operating in) server mode. As part of this, the language processor shall define a mapping from the LIPC model to its own procedure calling model.

NOTES

1 It is also possible in principle for a client processor to use the model for procedure calls defined in this International Standard to call procedures in the same language; executing on a server processor in the same language, and if the processor conforms in both client and server mode, it is even possible for it to serve itself using this model.

2 If a procedure is to be portable between processors which conform in server mode and the procedure is still to be called by client processors and programs, the procedure, and the processors, will also need to conform to the relevant language standard and the relevant standards binding for that language to the LIPC and LID standards.

5 A model of procedure calling: informal description

5.1 Model overview

A procedure is defined to be a closed sequence of instructions that is entered from, and returns control to, an external source.

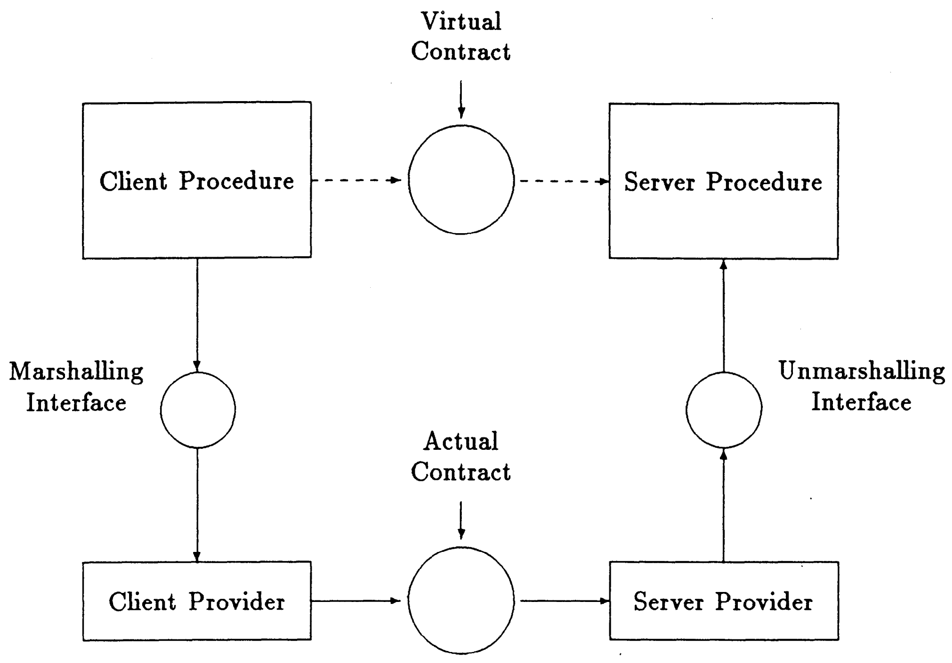
The general structure of a procedure call can be described as a single thread of execution in a particular program where the flow of control is passed from one procedure to another. The originator of the call is known as the client procedure and the procedure being called is referred to as the server procedure.

NOTE 1 – It is possible for a server procedure to also be a client procedure if it makes a call to another procedure in order to complete its desired function.

Procedures have the ability to exchange data between the client and server via the use of parameters (see 5.2). In addition, client and server procedures may also share data through the use of global data (see 5.2.2). In order for the parameters specified by the client procedure to be interpreted correctly, the parameters are required to be marshalled (see 5.2.3) to a base form for transmission that is shared by both the client and the server procedure. After the data has been transmitted, the server procedure must then unmarshall (see 5.2.3) the data from the base form into datatypes that are defined in the server language or in the language binding to ISO/IEC 11404 - Language-Independent Datatypes for that particular language.

NOTE 2 – An example of the process of marshalling and unmarshalling of parameters would be if a Pascal client procedure made a call to a Fortran server procedure passing a single character parameter by value. The Pascal "char" datatype would map to a LID character. In order to have the LID character be transmitted to the server procedure, the LID character is marshalled to an appropriate ASN.1 value, for example, which is a form that would be understood by both the client and server procedures. The ASN.1 value would then be transmitted to the server and upon receipt it is unmarshalled into a LID character, which in turn maps to a "character*1" in Fortran.

The following diagram outlines the basic components of the language-independent call model:



Language-Independent Procedure Call Model

This model illustrates how the client and server procedures communicate when their implementations conform to this International Standard. The virtual contract between the client procedure and server procedure is defined by the Interface Definition Notation contained within this International Standard. Upon the instantiation of a call, the marshalling interface marshalls the parameters and passes this information on to the client LIPC provider. The client LIPC provider is connected to the server LIPC provider via the actual contract which is the transmissible form (e.g., ASN.1). The server LIPC provider then unmarshalls the data, via the unmarshalling interface, into a form that is compatible with the server procedure. Upon return, the process is reversed with the unmarshalling interface now being the marshalling interface and the marshalling interface now being the unmarshalling interface.

5.2 Parameter passing

Any datatype defined in ISO/IEC 11404 - Language-Independent Datatypes can be the datatype of a formal parameter of a language-independent procedure call. This International Standard defines parameter passing solely on the passing of values. Therefore an actual parameter is any value of the datatype required by the call. The parameter passing model defined in this International Standard is a strongly typed model.

NOTE 1 – Weak typing can be accomplished by relaxing association rules and adding implicit datatype conversions in the language bindings to this International Standard.

The following notes relate the common parameter passing mechanisms that are found in existing languages to the four defined parameter passing schemes that are defined in this International Standard.