TECHNICAL REPORT

ISO/IEC TR 10182

First edition
2016-03-15

Information technology — Programming languages, their environments and system software interfaces — Guidelines for language bindings

Technologies de l'information — Langages de programmation, leurs environnements et interfaces logicielles des systèmes — Techniques d'interface pour les normes de langages de programmation

Reference number
ISO/IEC TR 10182:2016(E)

© ISO/IEC 2016



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2016, Published in Switzerland

All rights reserved. Unless otherwise specified, 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
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword ... **iv**

Introduction ... **v**

1 Scope ... **1**

2 Terms and definitions ... **1**
2.1 Terms ... 1
2.2 Abbreviated terms ... 3

3 Overview of functional binding methods ... **3**
3.1 Introduction to Methods ... 3
3.2 System Facility Standard Procedural Interface (Method 1) ... 4
3.3 User-Defined Procedural Interfaces (Method 2) ... 5
3.4 Programming Language Extensions with Native Syntax (Method 3) ... 5
3.5 Programming Languages with Embedded Alien Syntax (Method 4) ... 6
3.6 Binding Pre-Existing Language Elements (Method 5) ... 6
3.7 Conclusions ... 6

4 Guidelines ... **7**
4.1 Organizational Guidelines for Preparation of Language Bindings ... 7
4.2 General Technical Guidelines ... 9
4.3 Recommendations for Functional Specifications ... 9
4.4 Method-Dependent Guidelines for Language Bindings ... 10
4.4.1 Introduction to Method-Dependent Guidelines ... 10
4.4.2 Guidelines for Standard Procedural Interfaces ... 10
4.4.3 Guidelines for User Defined Procedural Interfaces ... 17
4.4.4 Guidelines for Programming Language Extensions with Native Syntax ... 18
4.4.5 Uses of Programming Languages with Embedded Alien Syntax ... 18

5 Future directions ... **18**

Annex A (informative) **Graphic Binding Examples** ... **20**

Annex B (informative) **GKS Bindings Generic Issues** ... **27**

Bibliography ... **40**

© ISO/IEC 2016 – All rights reserved

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 meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT), see the following URL: Foreword — Supplementary information.

The committee responsible for this Technical Report is ISO/IEC JTC 1, *Information technology,* Sub-Committee SC 22, *Programming languages, their environments and system software interfaces.*

This first edition of ISO/IEC TR 10182:2016 cancels and replaces the first edition of ISO/IEC TR 10182:1993, of which it constitutes a minor revision with the following changes:

— the references section has been deleted;

— minor editorial errors have been corrected.

 © ISO/IEC 2016 – All rights reserved

Introduction

This Technical Report is a compilation of the experience and knowledge gained by the members of ISO/IEC JTC1/SC22/WG11 (Techniques for Bindings) from the generation of programmers' interfaces to FUNCTIONAL INTERFACE STANDARDS. Although current experience was derived from the fields of computer graphics and database management, the problems discussed are thought to be generally applicable for mappings of other functional interface standards to programming languages. This Technical Report is intended

a) to identify the problems and conflicts which shall be resolved;

b) to suggest guidelines for future use;

c) to provide scope and direction to required additional work, such as common procedural calling mechanisms and data types; and

d) as a historical record of past experiences and decisions.

This Technical Report is incomplete; the authors have concentrated on those areas where experience and expertise was readily available. The ideas and issues brought forward here emerged from more than 10 years of work, and are represented in International Standards.

Clause 3 of this Technical Report contains the results of a survey of current methods used for language binding development. Characteristics of each method are given, followed by reasons for the selection of the method.

Application of the methods has suggested some guidelines that are presented in Clause 4. Clauses 3 and 4 contain documentation of the current state of language binding efforts; Clause 5 addresses future directions for language bindings.

Circulation of this Technical Report is necessary at this stage, as input and discussion from representatives of ISO/IEC JTC1/SC21 (functional specification standards developers), ISO/IEC JTC1/SC24 (computer graphics standards developers), and ISO/IEC JTC1/SC22 (language standards developers) is urgently sought. The Technical Report in its current form may be useful for those about to embark on language binding developments.

© ISO/IEC 2016 – All rights reserved

Information technology — Programming languages, their environments and system software interfaces — Guidelines for language bindings

1 Scope

This Technical Report is based on experience gained in the standardization of two major areas in information processing. One area covers programming languages. The other area is composed of the services necessary to an application program to achieve its goal. The services are divided into coherent groups, each referred to as a SYSTEM FACILITY, that are accessed through a FUNCTIONAL INTERFACE. The specification of a system facility, referred to as a FUNCTIONAL SPECIFICATION, defines a collection of SYSTEM FUNCTIONS, each of which carries out some well-defined service.

Since in principle there is no reason why a particular system facility should not be used by a program, regardless of the language in which it is written, it is the practice of system facility specifiers to define an 'abstract' functional interface that is language independent. In this way, the concepts in a particular system facility may be refined by experts in that area without regard for language peculiarities. An internally coherent view of a particular system facility is defined, relating the system functions to each other in a consistent way and relating the system functions to other layers within the system facility, including protocols for communication with other objects in the total system.

However, if these two areas are standardized independently, it is not possible to guarantee that programs from one operating environment can be moved to another, even if the programs are written in a standard programming language and use only standard system facilities. A language binding of a system facility to a programming language provides language syntax that maps the system facility's functional interface. This allows a program written in the language to access the system functions constituting the system facility in a standard way. The purpose of a language binding is to achieve portability of a program that uses particular facilities in a particular language. Examples of system facilities that have had language bindings developed for them are GKS, NDL, and SQL (see Bibliography). It is anticipated that further language binding development will be required. Some system facilities currently being standardized have no language bindings and additional system facilities will be standardized. There is a possibility of n × m language bindings, where n is the number of languages and m the number of system facilities.

The scope of this Technical Report is to classify language binding methods, reporting on particular instances in detail, and to produce suggested guidelines for future language binding standards.

Note that the language bindings and the abstract facility interfaces shall have a compatible run time representation, but the abstract facility does not necessarily have to be written in the host language. For example, if the application program is using a Pascal language binding and the corresponding facility is written in FORTRAN, there shall be a compatible run time representation in that operating environment. How this compatibility is achieved is outside the scope of these guidelines. This is generally a property of the operating environment defined by the implementor, and is reviewed briefly in this Technical Report.

2 Terms and definitions

2.1 Terms

ABSTRACT SERVICE INTERFACE: An interface having an abstract definition that defines the format and the semantics of the function invoked independently of the concrete syntax (actual representation) of the values and the invocation mechanism.

© ISO/IEC 2016 – All rights reserved

ALIEN SYNTAX: Syntax of a language other than the host language.

EMBEDDED ALIEN SYNTAX: Statements in a special language for access to a system facility, included in a source program written in a standard programming language.

EXTERNAL IDENTIFIER: An identifier that is visible outside of a program.

FUNCTIONAL INTERFACE: The abstract definition of the interface to a system facility by which system functions are provided.

FUNCTIONAL SPECIFICATION: The specification of a system facility. In the context of this Technical Report, the functional specification is normally a potential or actual standard. For each system function the specification defines the parameters for invocation and their effects.

HOST LANGUAGE: The programming language for which a standard language binding is produced; the language in which a program is written.

IDENTIFIER: Name of an object in an application program that uses a system facility.

IMPLEMENTATION-DEFINED: Possibly differing between different processors for the same language, but required by the language standard to be defined and documented by the implementor.

IMPLEMENTATION-DEPENDENT: Possibly differing between different processors for the same language, and not necessarily defined for any particular processor.

IMPLEMENTOR: The individual or organization that realizes a system facility through software, providing access to the system functions by means of the standard language bindings.

LANGUAGE BINDING OF *f* TO *l* or *l* LANGUAGE BINDING OF *f*: A specification of the standard interface to facility *f* for programs written in language *l.*

LANGUAGE COMMITTEE: The ISO technical Subcommittee or Working Group responsible for the definition of a programming language standard.

PROCEDURAL BINDING (system facility standard procedural interface): The definition of the interface to a system facility available to users of a standard programming language through procedure calls.

PROCEDURAL INTERFACE DEFINITION LANGUAGE: A language for defining specific procedures for interfacing to a system facility as used, for example, in ISO 8907:1987, Database Language NDL.

PROCEDURE: A general term used in this Technical Report to cover a programming language concept which has different names in different programming languages — subroutine and function in FORTRAN, procedure and function in Pascal, etc. A procedure is a programming language dependent method for accessing one or more system functions from a program. A procedure has a name and a set of formal parameters with defined data types. Invoking a procedure transfers control to that procedure.

PROCESSOR: A system or mechanism that accepts a program as input, prepares it for execution, and executes the process so defined with data to produce results.

PROGRAMMING LANGUAGE EXTENSIONS WITH NATIVE SYNTAX or native syntax binding: The functionality of the system facilities is incorporated into the host programming language so that the system functions appear as natural parts of the language. The compiler processes the language extensions and generates the appropriate calls to the system facility functions.

SYSTEM FACILITY: A coherent collection of services to be made available in some way to an application program. The system facility may be defined as a set of discrete system functions with an abstract service interface.

SYSTEM FACILITY COMMITTEE: The ISO technical subcommittee or Working Group responsible for the development of the functional specification of a system facility.

© ISO/IEC 2016 – All rights reserved

SYSTEM FUNCTION: An individual component of a system facility, which normally has an identifying title and possibly some parameters. A system function's actions are defined by its relationships to other system functions in the same system facility.

2.2 Abbreviated terms

CGI: Computer Graphics Interface standard (ISO DP) — a functional specification of the computer graphics programming system facility.

GKS: Graphical Kernel System standard (ISO/IEC 7942-1) — a functional specification of the computer graphics programming system facility.

MDL: Module Definition Language — a language for the specification of an interface to a generic system facility. The MDL is used to generate a module to support the specific system facility access needs of an application program.

NDL: Network Database Language — may be used to define the structure of a database using the network model of data. NDL is defined in ISO 8907:1987 (see Bibliography). The standard also includes the data manipulation functions and their language bindings.

PHIGS: Programmers Hierarchical Interactive Graphics System standard [ISO/IEC 9593 (all parts)], – a functional specification of the 3-D computer graphics programming system facility.

SQL: Structured Query Language — defines the structure of a database using the relational model of data. Database Language SQL is defined in ISO/IEC 9075-1, ISO/IEC 9075-2, ISO/IEC 9075-3, ISO/IEC 9075-4 and ISO/IEC 9075-11 (see Bibliography). The standard also includes the data manipulation functions and their language bindings.

3 Overview of functional binding methods

3.1 Introduction to Methods

This section discusses the binding development problem in general by documenting a number of different approaches to bindings. Each approach has its own characteristics from the points of view of the user, the implementor, and the specifiers of standards.

The first task in specifying a binding of a system facility is to determine the usability, stability, and implementation goals of both the binding and the system facility, and to use these to help select the best method.

The functional binding methods are:

— Method 1. Provide a completely defined procedural interface (the System Facility Standard Procedural Interface).

— Method 2. Provide a procedural interface definition language (User-Defined Procedural Interface).

— Method 3. Provide extensions to the programming language, using native syntax.

— Method 4. Allow alien syntax to be embedded in the programming language.

— Method 5. Binding pre-existing language elements.

Before addressing the individual methods, a discussion of a general issue that affects programming language implementations is indicated. This issue is whether to increase the capability of a given compiler to encompass the system facility, or to provide a pre-processor. Though this is of no direct concern to language binding developers, they may wish to consider the feasibility of each option when choosing a method.

© ISO/IEC 2016 – All rights reserved