TECHNICAL REPORT

ISO/IEC TR 11017

First edition
1998-03-01

Information technology — Framework for internationalization

Technologies de l'information — Cadre pour l'internationalisation

Reference number
ISO/IEC TR 11017:1998(E)

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Committee) 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 main task of technical committees is to prepare International Standards, but in exceptional circumstances a technical committee may propose the publication of a Technical Report of one of the following types:

– type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

– type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

– type 3, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

ISO/IEC TR 11017, which is a Technical Report of type 3, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

Introduction

Background

ISO/IEC JTC 1 created SC 22/WG 20 with a specific task to address internationalization in programming languages and applications. The development of a Technical Report for a framework for internationalization of programming languages and applications was selected as the first project of the working group. The update of ISO/IEC TR 10176:1991 "Information technology — Guidelines for the preparation of programming language standards", was added to the projects of the WG 20 to ensure that the other working groups in the SC 22 profit from WG 20's work by including the services described in the framework document, into the development and update of programming language standards.

After surveying the various requirements, ISO/IEC JTC 1 SC 22 defined a set of services that might be provided in a truly internationalized application, and thus also in programming languages. These services are described in this Technical Report, together with short explanations of the requirements and a list of standards on which they are based. This Technical Report forms the base document for the consideration of internationalization features in applications.

The revision of ISO/IEC TR 10176, introduces the services defined in this Technical Report into the development of programming language standards. The recommendations in the guide will enable programming languages to deal with the issues of internationalization and to create programs that can run in culturally different environments without changes to the program code.

While preparing this Technical Report, it became obvious that the culturally correct ordering (sorting) of data is an essential part of most commercial processing. This is particularly difficult when the underlying character set is capable of representing all of today's written languages. SC 22 therefore proposed, and was assigned, an additional work item to develop a standard for "Information technology — International String Ordering — Method for Comparing Character Strings and Description of Default Tailorable Ordering", ISO/IEC 14651.

"The specification of cultural conventions in language independent form" is another work item assigned to SC 22. Programming language specific specifications for such repositories exist in ISO/IEC 9945-2 and in publicly available specifications from various consortia. It is a requirement that output from this work (ISO/IEC 14652) must be compatible with the work already done by ISO/IEC JTC 1 SC 22/WG 15 (POSIX working group) in this field. A working draft exists for this Technical Report.

This Technical Report

The role of information technology in daily life is rapidly growing, and its prevalence now approaches that of social infrastructures such as water or power supplies, public communications links and road or public transportation systems. To ensure the future success of this technology, all communication between systems and users of the systems must be in the natural language of the user in the user's script, and must fit the natural behaviour of the user.

Traditionally, the process of the adaptation to a user culture has been called the internationalization (or localization, or customization in an intermixed way) of information technology.

Historically, the internationalization of information technology has been provided on a demand/requirement basis, and thus solutions have been based on the "best available technology" approach. As a result, the solutions to internationalization for different systems do not necessarily share any common directions or goals. In many cases, different technology solutions have different goals altogether.

This Technical Report is intended to present an organised overview of, and relationship between, the various requirements, approaches and solutions to the internationalization of information technology systems. In addition, this Technical Report recommends services for internationalization of software.

This Technical Report refers to the ISO/IEC JTC 1 TSG-1 final report as a base document (see Annex A).

The purpose of the ISO/IEC JTC 1 internationalization activity is to establish the goals and directions for its production of applications adapted for use in the international environment which can be customised to local cultural conventions, and then to produce the necessary standards. The common internationalization standards will benefit a number of parties in the information technology field:

— users will have a much greater chance of having applications which are compliant with their local cultural conventions, and will also be able to experience a much greater commonality between the handling of cultural conventions in different applications.

— application developers will be relieved from specifying a number of internationalization parameters, which can simply be taken from ISO/IEC member bodies specifications.

— procurers can specify cultural convention sets and obtain well-defined uniform behaviour across hardware and software platforms.

— ISO/IEC member bodies can produce specifications for their national cultural conventions, which then can be applied uniformly across application platforms.

— ISO/IEC JTC 1 working groups can produce bindings to ISO/IEC JTC 1 standards that use the specifications of cultural conventions from ISO/IEC JTC 1 member bodies, and thus reuse internationalization data and specifications without further work.

As the internationalization activity involves many parties, it is planned to have a staged approach to producing the relevant standards. First a framework document will be produced (this document); then the "Guidelines for the preparation of programming language standards" will be revised with respect to internationalization (a revised ISO/IEC TR 10176); and finally the various internationalization standards will be produced.

This Technical Report is structured as follows:

Clause 1 describes the scope of this Technical Report, clause 2 lists the documents referenced, and clause 3 defines the key terminology (Only those items necessary for the understanding of this Technical Report are referenced in clause 2 and clause 3).

Clause 4 presents many different categories of internationalization, all of which are useful in internationalized information systems.

Clause 5 recommends the INTERNATIONALIZATION/LOCALIZATION model as a comprehensive solution to accommodate the different types of internationalization described in clause 4.

Clause 6 itemises the major conventions which differ between cultures, also emphasizing that number of these conventions are open ended.

Clause 7 presents the basic model for internationalized systems and recommends a set of services which are necessary to support the major cultural differences (described in clause 6) based on the model.

Annexes provide supporting information to the main text.

Information technology — Framework for internationalization

1 Scope

This Technical Report presents the framework and recommended model for internationalization and identifies the services required for the internationalization of information technology.

This Technical Report:

— discusses the requirements of internationalized systems and their users,

— suggests a concerted, unified approach to internationalization,

— recommends services to fulfil the requirements, and lists the standards related to the services.

This Technical Report is intended to be a reference for future standards on the internationalization of information technology, to act as a communication vehicle between those who provide standards and those who request them, and also to function as a basis for all future ISO/IEC JTC 1/SC 22/WG 20 activities.

NOTE 1 This Technical Report does not propose any specific solutions for internationalization. Future standards are expected to provide the solutions, reflecting the directions in this Technical Report.

NOTE 2 Internationalization services consist of internationally generic services and nationally specific information (data). This Technical Report covers only the internationally generic portion of software, so it does not discuss:

— preparation of internationalized information technology systems for localization by a local user.

— hardware-related issues or requirements.

— requirements related to system ergonomics.

NOTE 3 The internationalization solutions and services available in the mid-late 1980s are the technical basis for this Technical Report.

2 References

See annex A.

3 Definitions

For the purposes of this Technical Report, the following definitions apply.

3.1 application platform:
A set of resources on which an application will run.

3.2 application program interface (API):
The interface between the application software and the application platform, across which all services are provided.

3.3 byte:
A bit string that is operated as a unit.

3.4 character repertoire:
A specific set of characters that are represented in a coded character set.

3.5 coded character set:
A set of unambiguous rules that establishes a character set and the relationship between the characters of the set and their coded representation.

3.6 cultural convention:
A convention of an information system which is functionally equivalent between cultures, but may differ in presentation, operational behaviour or degree of importance.

3.7 cultural convention set:
A set of cultural conventions for which support services are defined to be ideally located within the application platform. (Services are normally available through Application Program Interfaces [APIs])

3.8 INTERNATIONALIZATION (I18N):
A process of producing an application platform or application which is easily capable of being localized for (almost) any cultural environment. (Note, therefore, that an INTERNATIONALIZED information system does not have a dependency on any specific culture unless it is LOCALIZED to that selected culture.)

3.9 LOCALIZATION (L10N):
A process of adapting an INTERNATIONALIZED application platform or application to a specific cultural

environment. In LOCALIZATION, the same semantics are preserved while the syntax may be changed (Refer to clause 5).

3.10 octet:

An ordered sequence of eight bits considered as a unit.

3.11 ordering:

An operation by which two different objects (for example two character strings) are assigned a context free deterministic order.

3.12 script:

A set of graphic characters used for the written form of one or more languages.

4 Categories of user culture for internationalization

Internationalization is the adaptation of a user-interface to a user culture. However, it is a fact that there are many different views on user culture. Each user culture may define its own requirements for internationalization which are different from each other. This clause gives an overview of those user culture differences which are needed to understand the requirements for internationalization.

4.1 User interfaces

Applications must be user-friendly. A user interface containing familiar terms is of paramount importance to user-friendliness.

The user interface needs to ensure meaningful input and output for the user. Messages and representation of cultural conventions such as dates and times should be provided in a language and format which is natural to the user. Incorporating these capabilities into a system increases user acceptance and decreases error rates in human-machine interactions.

The requirement for this user-friendliness to support a wide range of cultures in the world is called the "need for the internationalization of information technology" in a general sense.

4.2 Scope of user culture

The scope of user interface applicability varies from person to person. An application which is to be implemented in several different countries or cultures will clearly need to be implemented in such a way that it can provide suitable output for, and accept suitable input from, its different users. Even applications produced for a single country or a single location may still need to accommodate users with different cultural backgrounds within that environment; for example Canada, Belgium, European Union Secretariat or airports.

The requirements for user-friendliness can vary widely depending on the combinations of cultural adaptation, languages, scripts etc.

This subclause presents a basic classification of the variations. Note that all of the requirements described in 4.2.1 to 4.2.7 are just one aspect of the overall internationalization requirement.

4.2.1 Mono-culture/script/language

This is the most simple case of user-friendliness, and the most primitive and basic requirement for internationalization.

The requirement is for:

— Users to be able to read and write one language using one script.

— All users to have a single cultural background.

4.2.2 Mono-culture/script/language, including ISO/IEC 646

A user interface that offers a mono-culture/script/language is regarded as normal, and the basic requirement for user-friendliness. However, the history of information technology shows that ISO/IEC 646-IRV, a national version, or a similar character set, is also necessary in most cases in addition to the basic native script. In practice, therefore, the coded character set which supports the native script of the user might be expected to include the repertoire of the ISO/IEC 646-IRV coded character set, if it does not already include it.

The requirement is for:

— Users to be able to read and write one language using their own native script, which may include the base Latin alphabet of ISO/IEC 646.

— All users to have a single cultural view.

— Some (or most) system-level user interfaces and programming languages to use a "computer language" written in a "computer repertoire" based on the Latin alphabet (ISO/IEC 646).

This is the basic requirement, and the most widely accepted user interface applicability of internationalization. This is often called a bi-lingual system, but in reality, it is a system with dual coded character sets.

4.2.3 Mono-culture/language, Multi-script

The extension of the mono-culture/script/language requirement, including ISO/IEC 646, to allow additional scripts is a possible user requirement. In this Technical Report, this is called a Multi-Script system. Note that this refers to a single language and culture; embedded French or Spanish text in a Korean document for Korean user in Korean language and culture might be an example of this type of application.

4.2.4 Mono-culture, Multi-script/language

The extension of the mono-culture/language, multi-script requirement to multiple languages is a natural extension to the above.

Some countries or regions share almost similar cultural requirements with an acceptable degree of compromise, even if the languages and scripts are different. Sometimes unification of these cultural aspects will satisfy the user needs for this type of internationalization requirement. Some of the pan-European systems which support "Western culture" and several European language might be a sample of this type of multi-lingual system.

There are two types of multi-lingual/script mono-culture systems. One is the sequential and the other is the concurrent type. (see 4.2.5.1 and 4.2.5.2 for a description of these two types in the multi-cultural case.)

4.2.5 Multi-culture/script/language

Finally, multiple solutions for each components may be required. In this Technical Report this is termed a multi-lingual system. In practice, multi-lingual systems can be classified into two groups:

4.2.5.1 Sequential multi-lingual

A sequential multi-lingual system is able to service several kinds of users, but only one kind of user at a time.

4.2.5.2 Concurrent multi-lingual

A concurrent multi-lingual system is able to service several kinds of users as if the services are provided simultaneously from the human user's point of view.

4.2.6 Global Uniformity

Global Uniformity aims to minimize the information ambiguity due to the culturally dependent multiple solutions for the same items of information. It also aims to maximize the scope of user interfaces and the acceptance of international standards. Simultaneously, the number of necessary interfaces decreases, and conventional ambiguities, such as date format, disappear. (For example, a date format of 01-02-03 means February 3, 2001, January 2, 2003, or February 1, 2003 depending on cultural conventions. For Global Uniformity, one standard format would be chosen, thus eliminating ambiguity.)

NOTE 4 Historically, ISO and IEC have pursued this Global Uniformity strategy with respect to other technologies. In the field of information technology, however, multiple views of internationalization and international standards are also necessary.

4.2.7 Cross-Cultural Friendliness

Like Global Uniformity, Cross-Cultural Friendliness aims to maximize user interface applicability and user-friendliness. In this context, "friendliness" denotes the ease with which unfamiliar culturally-dependent information can be understood by persons who are not familiar with this culture. This is accomplished through strategic adaptation of cultural conventions within one culture which appear familiar to new users. As the popularity of information technology increases, the frequency of confrontations with unfamiliar data increases, thus justifying the need for Cross-Cultural Friendliness.

Cross-Cultural Friendliness supports users who are unfamiliar with the culture by widening the overall applicability of cultural conventions. For example, Spanish dictionaries order CH after CZ, which contrasts with the collation sequence of most of other Latin script based languages. Users unfamiliar with the Spanish alphabet might assume an order consistent with the system already known from other languages; users familiar with the Spanish alphabet would not, however, feel comfortable with the ordering system provided for users who are unfamiliar with Spanish (CH after CG, before CI). Another example of this kind of situation occurs in Denmark, where AA is ordered as one single letter at the end of the alphabet.

Another example of Cross-Cultural Friendliness is the input method for unfamiliar characters. The CJK (Chinese, Japanese, Korean) ideographic input method for users who are not familiar with the CJK ideographs may not be acceptable to Chinese, Japanese or Korean natives, but it might be easy to use for those who are infrequent users of CJK ideographs.

Cross-Cultural Friendliness is, therefore, one of the aspects of internationalization considered in this Technical Report.

4.2.8 User-Culture Summary

In summary, a "friendly" system could mean one of the following:

— a system which supports one of many different cultures (that is, "friendly" to one culture);

— a system which supports one of many different cultures (that is, "friendly" to one culture) as well as supporting ISO/IEC 646-IRV coded character set;

— a system which supports several scripts simultaneously within one culture and one prime language;

— a system which supports multi-script/ language sequentially in one culture;

— a system which supports multi-script/ language concurrently in one culture;

— a system which supports multi-culture/script/ language sequentially;

— a system which supports multi-culture/script/ language concurrently;

— a system which supports Global Uniformity;

— a system which supports Cross-Cultural Friendliness;

4.3 A Future Vision

The needs of internationalization expressed in 4.2.1 to 4.2.7 are necessitated by current information technology. Inter-personal communication via information technology has traditionally been confined to text and keyboard applications within machines in a single culture. The requirements and standards for internationalization are highly volatile. Communication media are broadening to include multi-media data, and information technology is growing toward an age of general machine-facilitated communication between humans from diverse backgrounds.

The required technology components for message passing between cultures must first be present at both ends of a communication link. The people communicating must also have knowledge about the conversation topic, and must also have an adequate understanding of all the cultures involved in the discussion. Only then can the true inter-cultural communication take place directly through the machine. Each user can interact with the machine in a different natural language, and the machine will convert the information into the appropriate form for the recipient. This is the future vision of the internationalization of information technology.

The issue of the requirement for support services for this future vision of internationalization is not addressed in this Technical Report, because the technology to implement this future vision is not yet widely and reliably available.

5 INTERNATIONALIZATION and LOCALIZATION

The requirement for the integration of script, natural language and local practice into information systems is additional to basic system requirements. Such integration is a user requirement and is thus an "fundamental requirement", because it constitutes a basic constraint which the system must satisfy. If the system meets all fundamental requirements and is sufficiently user-friendly, the methodology which ensures this "friendliness" is not a primary concern for its users.

However, this methodology does concern suppliers whose goal is to meet user requirements efficiently and economically. INTERNATIONALIZATION facilitates the provision of similar applications to different cultures with maximum economic gain, efficiency and quality. Although numerous approaches exist for providing local adaptation solutions, each faces similar initial problems and cultural differences.

This clause presents an INTERNATIONALIZATION/ LOCALIZATION approach as the recommended methodology of this Technical Report.

5.1 Local adaptation

Providing "User friendliness" to the end user is part of the adaptation of an information system to local needs. There are very many varieties of "locals" and many varieties of "friendliness", as described in 4.2.1 to 4.2.7 and it is possible to respond to any of localization requirement by optimising the technology to suit that specific requirement. However, meeting the requirements of all combinations individually is impractical in terms of resources. There is a need for a basic universal solution to meet the fundamental needs of all cases, such that the total resources required for the local adaptation of information technologies to specific local needs may be minimised.

NOTE 5 The end user should see no difference between the results of specific local adaptations and the results of applying a uniform and universal methodology. What end users are looking for is simply friendliness and familiarity to them.

5.2 Current local adaptation approach

Many system developers incorporate into an application the hard-coding necessary to support one cultural environment. That application then requires substantial rewriting to support other cultural environments. Thus, programs with identical functions are developed repeatedly. Not only does this waste design and programming effort, but it creates the following problems:

— Application development is expensive because code is rewritten for different cultures;

— High cost inhibits many applications from being developed for multiple environments, which automatically limits the potential market of those applications;

— Justification of the cost of the local adaptation for minorities is very difficult. Sometimes, this fact unfairly penalises the requirement of small markets;

— The rewriting of code results in staggered marketplace introductions of an application. This may cause serious problems with some applications, such system software distributed in world-wide networks;

— The possibility of application inconsistencies in different cultures arises even though the external functional interfaces may be identical. This creates problems in world-wide networks and leads to updating/maintenance complications such as inconsistent next-generation system upgrading.

— Ownership of rewritten applications becomes unclear, so maintenance and support responsibilities become lost.

For these reasons, alternative systematic approaches are required to make application development more efficient and consistent.

5.3 The INTERNATIONALIZATION/LOCALIZATION approach

INTERNATIONALIZATION/LOCALIZATION simplifies the development of systems for different cultural environments and substantially reduces the necessary production efforts. It also facilitate the uniform addition of cultural features to all applications.

INTERNATIONALIZATION is intended to permit the design and implementation of an application platform and/or an application which can accommodate users with a variety of cultural backgrounds. Services are required which insulate the application platform and/or the application from a variety of cultural differences that are not relevant to its functionality.

A system which provides this service is referred to as an INTERNATIONALIZED system in this Technical Report.

5.3.1 INTERNATIONALIZATION

An INTERNATIONALIZED system has a distinct function, but its code contains predetermined slots into which cultural information will be loaded. This cultural information relates to the cultural conventions and messages in the natural language(s) and script of the user of the system, and allows the user to communicate with the system in familiar terms and in familiar forms. Ideally, information from any culture can be fitted into any system framework, thus changing the system user interface but maintaining the functionalities of the system unchanged. In other words, the system is free of any culturally dependent items.

INTERNATIONALIZATION is the process of constructing such a system framework. Once the INTERNATIONALIZED system is associated with a specific cultural information and system platform (a computer or an operating system), the entire system is called a LOCALIZED system. Such an approach can satisfy the needs of users from different cultures without requiring modifications to system functions or application software.

5.3.2 LOCALIZATION

LOCALIZATION is the adaptation of INTERNATIONALIZED systems to the specific cultural needs of users. This occurs when cultural information is fitted into the predetermined slots in the system framework. Thus, the cultural data is the only component of an INTERNATIONALIZED system that changes, and this change occurs only to suit the culture of the user. The functions and basic coding of the system always remain same.

LOCALIZATION data may be provided not only for specific single cultures, but also for multiple cultures. In fact, all culture combinations described in 4.2.1 to 4.2.7 may be loaded into an INTERNATIONALIZED system as part of the LOCALIZATION process. Even Global Uniformity and Cross-Cultural Friendliness are only examples of single cultural data sets for LOCALIZATION.

NOTE 6 It is not necessary to start the process of internationalization from the U.S. culture. Once an INTERNATIONALIZED system is in place, it can be LOCALIZED to the U.S. culture.

5.3.3 Summary

In summary, INTERNATIONALIZATION is "super-localizability" which ensures lower cost, higher quality, and faster delivery of LOCALIZED applications. All INTERNATIONALIZED applications must still be subsequently LOCALIZED, but the LOCALIZATION process is cost and time effective.

The pictures in FIGURE 1 symbolise the INTERNATIONALIZATION/LOCALIZATION approach. Once space for any language is provided, the girl can speak any language in any script — even a Global Uniformity solution is possible.

On the other hand, it is very difficult to carry out local adaptation when the message is hard-coded, as in FIGURE 2, without some or even significant degradation. Local adaptation from one culture to another is impossible in practice.

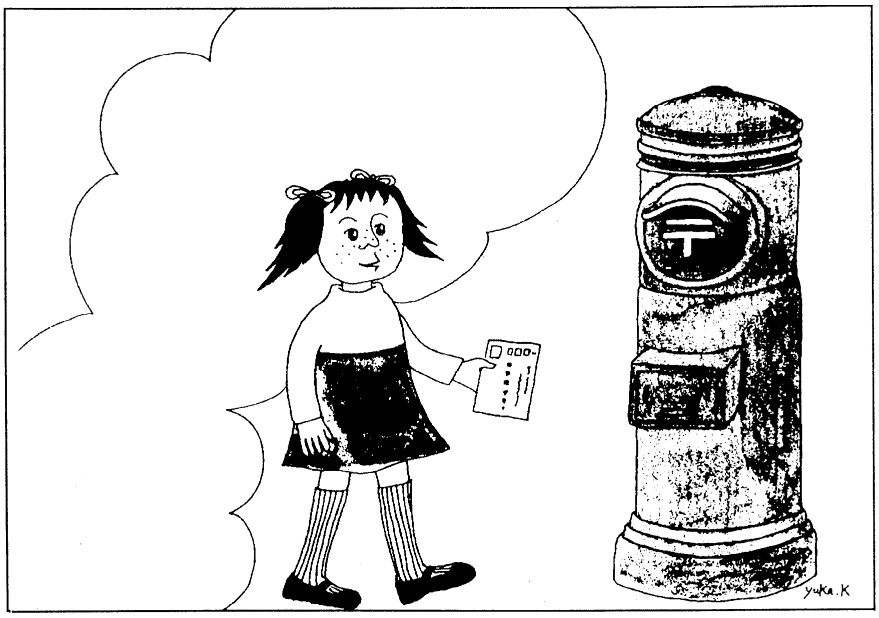
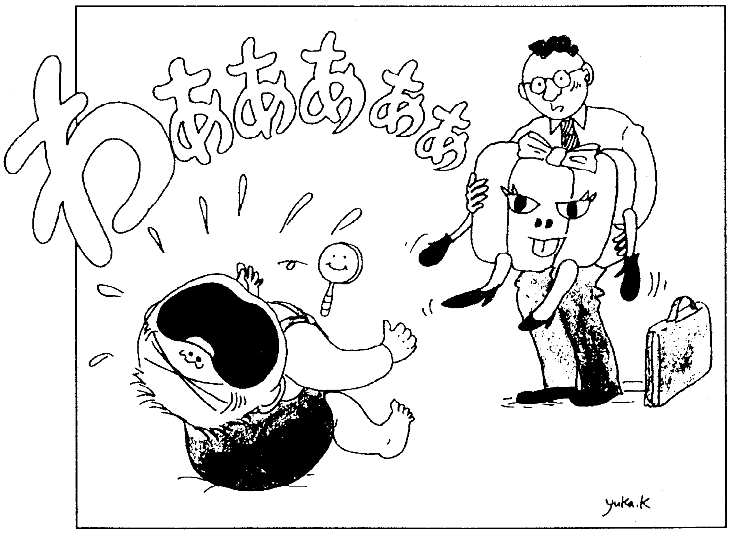


Figure 1 — INTERNATIONALIZED System free space for messages



1) hard coded message (low localizability) (Messages in Japanese characters are part of the picture.)

Figure 2 — Hard-coded localization

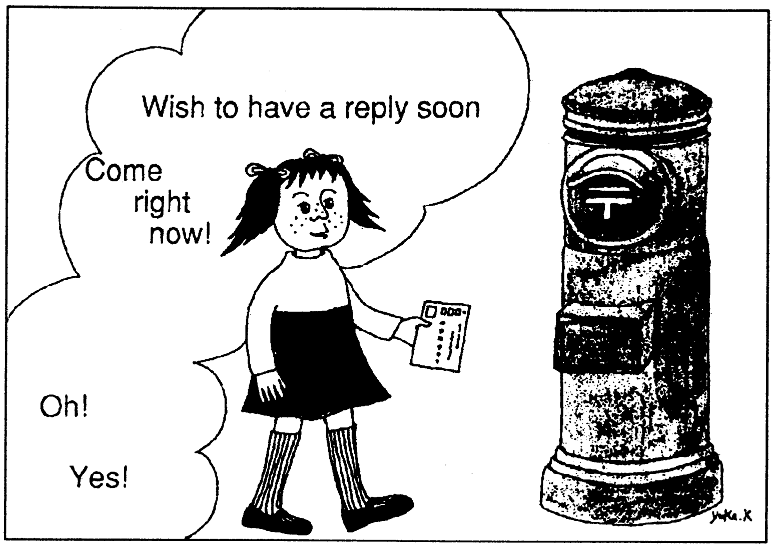


Figure 3 — Examples of a LOCALIZED English system

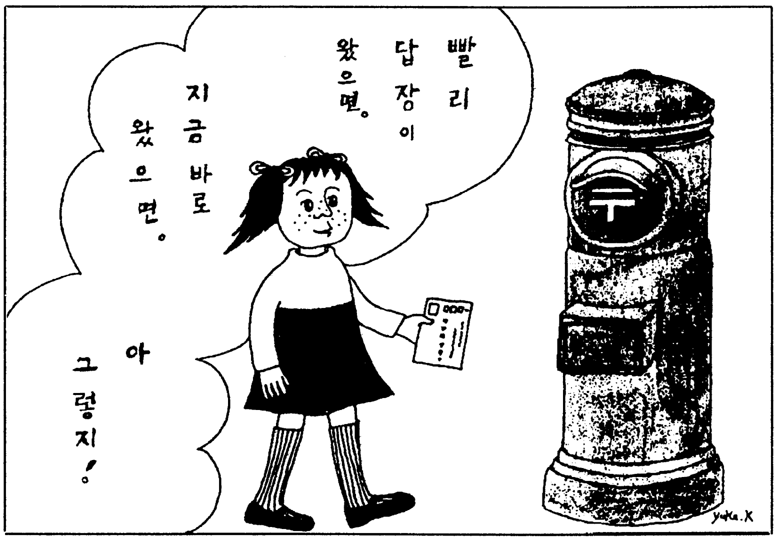


Figure 4 — Examples of a LOCALIZED Korean system

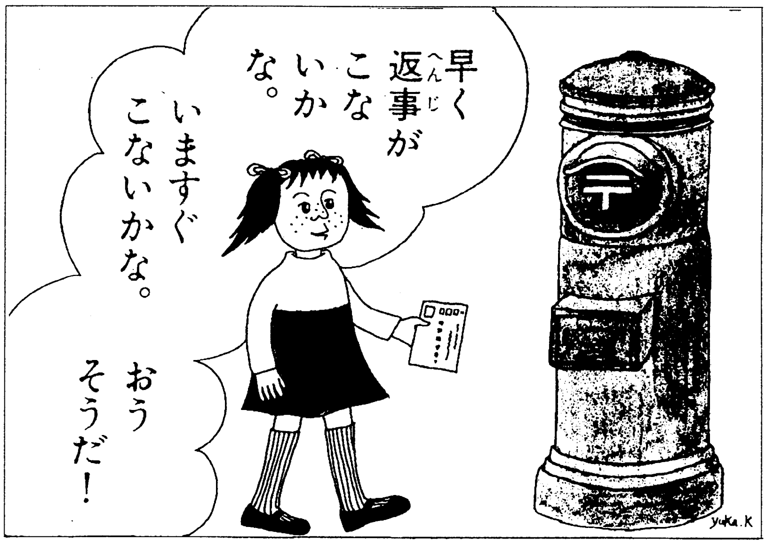
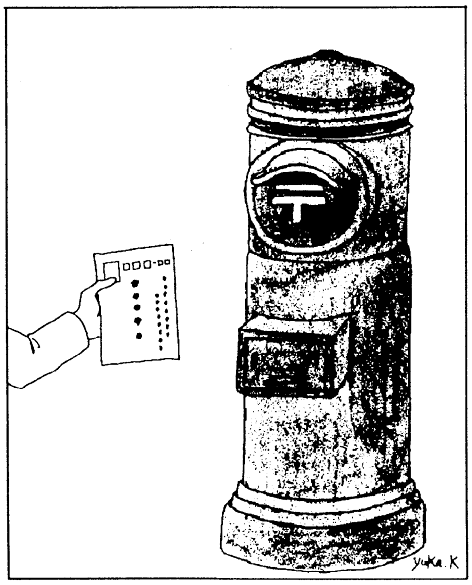


Figure 5 — Examples of a LOCALIZED Japanese system

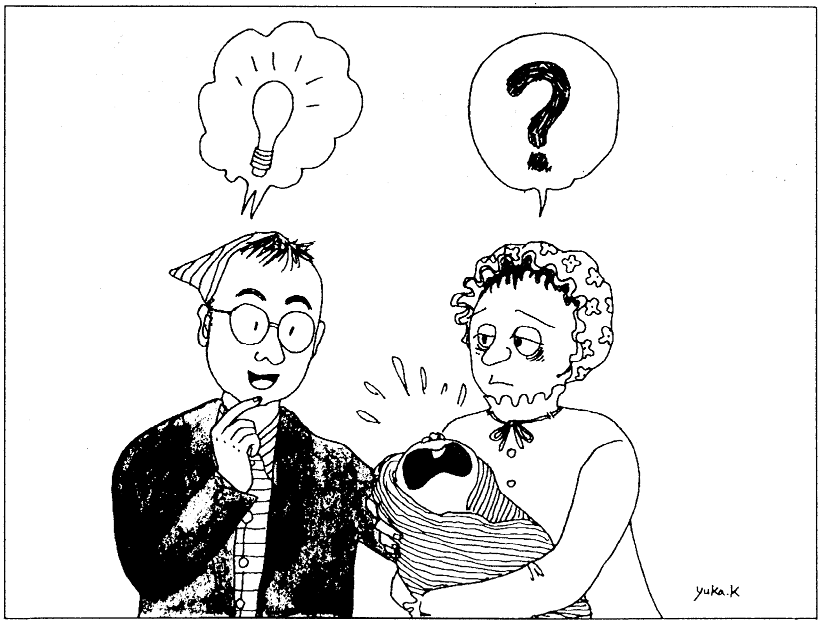
FIGURE 6 shows a sample of cultural conventions. Without adapting the cultural conventions to the target culture, the true user friendliness of the solution is lost to some degree.



2) The address format of mail and the "Post" mark on mailboxes are culture dependent.

Figure 6 — Cultural conventions to be localized

FIGURE 7 is a LOCALIZED example which is achieved by Global Uniformity. The question mark and light-bulb in the message space do not require translation to any culture (see 4.2.6).



3) The meaning of a question mark is understood in almost all cultures.

Figure 7 — Global Uniformity

5.4 Environments for internationalization

The requirements for "user friendliness" in different cultures are often facilitated by the introduction of new technologies, and new technologies encourage further requirements for user friendliness. TABLE 1 contrasts the traditional common practice with the needs of international environments for information technology.

Table 1 – Environments for internationalization

	Common Practice	**International environment**
coded character set	specific to each application platform (mostly ISO/IEC 646-IRV)	any coded character set
character repertoire	tightly linked with character code	code independent
one character	one byte	one or more bytes
one byte	7 or 8 bits	8 bits or more
display width of a character	one column on a character cell display	variable numbers of columns on a character cell display
	fixed width	variable width
character count	=byte count, =display width	not always equal to byte count, not always equal to display width
maximum number of a character	128 or 256 in most cases	unknown
printable character	ISO/IEC 646-IRV	no assumption can be made
collating sequence	ISO/IEC 646-IRV, per code sequence	no assumption can be made
input from keyboard	direct	specially designed device, or indirectly using(interactive) interpreting methods
message	in English	in any language
writing direction	left to right and top to bottom	might vary, even within one execution environment
coded character set of program source code	same as execution coded character set	an execution coded character set might be different from the coded character set of program source code
coded character set of file	coded character set of program	not always the same as coded character set of program

5.5 Relationship between INTERNATIONALIZATION and Application Portability

In principle, the environment of an application consists of a human user and the application platform. Usually, Application Portability indicates the ability of an application to run on different platforms without change to the program source code. To accomplish this, the Application Programming Interface (API) must be standardized for compatibility with different application platforms.

INTERNATIONALIZATION, on the other hand, effectively allows an application to be run for any user. However, only the user interface is modified for compatibility with different users, and the application platform (or the family of application platforms) remains same. In principle therefore, INTERNATIONALIZATION does not require the consideration of Application Portability across different application platforms.

In practice, Application Portability and INTERNATIONALIZATION may share some support functionality; thus, INTERNATIONALIZATION can be done while maintaining Application Portability, however, as described above, this is not a necessary condition.

Most applications only communicate with users through application platforms. So, applications are only useful when paired with platforms. Thus, application and platform should be INTERNATIONALIZED as paired set. This is another reason why INTERNATIONALIZATION should not be considered as being similar to Application Portability between application platforms.

5.6 What is culture? (Relationship to functionality for Customization)

There are many other customer requirements that can be categorised as cultural requirements. However, those functional requirements that do not stem from cultures, geographically and socially speaking, are not considered in detail in this Technical Report. In other words, this Technical Report addresses generic cultural requirements that are common among applications as well as platforms (clause 6.3 provides an initial list of those requirements).

There are a number of cultures for which functionality is not discussed in this Technical Report; for example, all organizations have their own habits, customs and methods, which are known collectively as organizational culture. Similarly, each profession has its own terminology and methods, which are seen as a culture of the specific profession. Traditionally, the adaptation of systems to cultures (or sub-cultures) such as organizational and/or professional cultures has been achieved by modifying systems at user sites. This provision of further cultural APIs for customer support is called Customization.

INTERNATIONALIZATION and LOCALIZATION are distinct from Customization even though they may share some features. Any system has first to be INTERNATIONALIZED in order to be adaptable to the different cultures and geographical requirements.

Normally, after INTERNATIONALIZATION, LOCALIZATION is first carried out for each culture to implement the generic cultural conventions specified for this culture, and then Customization takes place as the final step to cater for non-generic cultural needs that are relevant to a specific industry, profession or local area.

The LOCALIZATION and Customization processes need not always be in this sequence. For example, world-wide organizations need to Customize their information systems for their own operational cultures, and the Customized systems must be LOCALIZED to suit each local culture and location in which they are deployed. If the three layers mentioned were to be implemented simultaneously, considerable confusion would result for the support and maintenance of the systems. To avoid this, each layer of the adaptation work needs to be isolated as cleanly as possible.

In conclusion, INTERNATIONALIZATION/LOCALIZATION, Application Portability and Customization may look somewhat alike and may share some common features in the work necessary to achieve them, but as a matter of principle they must be considered and handled separately.

5.7 What is the cultural data to be loaded?

Clause 4.2.8 discusses the nine different kinds of user friendliness which are recognised as providing some measure of internationalization. However, only one approach, the INTERNATIONALIZATION/LOCALIZATION model, is recommended in this Technical Report.

This means that to support all aspects of user friendliness, one culturally independent solution is recommended, and all differences should be catered for by the data loaded into the INTERNATIONALIZED system. There may be a need to "load" single culture data into the INTERNATIONALIZED system to achieve a mono-lingual system solution, or to accommodate multiple culture data for multi-lingual environments.

The data load is, in principle, totally user-defined, given the international application. The data can vary across language, country, organizational and professional culture, application or other user preferences.

Occasionally, different cultures may exist within the same geographic location. For instance, some countries use two different scripts, along with associated cultural conventions, to represent their natural language; for example, Cyrillic and Arabic script may both be used to represent some Iranian and Turkic languages. Such differences are not an issue for INTERNATIONALIZED systems, but should be addressed by applying different cultural data to the same INTERNATIONALIZED system, perhaps simultaneously.

A Global Uniformity solution envisages the use of special data, having an agreed uniform meaning world-wide, within the INTERNATIONALIZED system. Global Uniformity can, therefore, be specified as a localization for the Global culture of an internationalized system.

5.8 Localization Data must be loaded in a flexible manner, allowing future growth and providing for competitive innovation

Extension of the principle of INTERNATIONALIZATION/LOCALIZATION opens the door to the possibility of new types of "friendliness" in information systems. Since the data to be loaded for LOCALIZATION is largely unrestrained (at least in theory), solutions should look beyond currently known techniques and provide the flexibility to accommodate (say) future easy-to-use interfaces, yet to be defined. INTERNATIONALIZATION should not miss such future opportunities, and the design of the "cultural vacancy" of the INTERNATIONALIZED system should be as independent as possible of the data to be loaded.

A good example might be the input method for Japanese characters, where there is still competition between new easy-to-use Japanese input methods, using the same interface to the system, users of information systems have benefited greatly from the availability of a standard interface and from the ongoing free competition between input methods. It follows that rigid optimisation of the design of the "vacancy" for Global Uniformity and/or Cross Cultural Friendliness data should be avoided.

6 Cultural differences

The requirement for user-friendliness has four key components:

— Communication with the system should be in the user's language.

— The user's language should be written using the user's familiar script.

— The data written in the user's script should be processed in a manner which is "culturally correct" for that user.

— Other behaviour of the system should also be in accordance with the user's culture.

The requirements for each of the components listed above differ from culture to culture. This clause describes the differences, itemising the various specific requirements of cultures (fundamental requirements), but not discussing the methodologies necessary to provide solutions for these requirements within a real system environment. Examples of such solutions are discussed in clause 7 of this Technical Report.

It should be noticed that a direct one-to-one relationship between a requirement (listed in this clause) and a specific solution for the requirement, samples in clause 7, is not necessary. Rather, fundamental requirements dictate the need for additional functionality (secondary requirements) to accommodate the fundamental requirements within the system. For example, a cultural environmental switching mechanism is not a user requirement, but it is a secondary requirement and may be one of the mechanisms necessary in order to fulfil the fundamental requirements.

6.1 Requirements for Cultural Dependencies

It is necessary to adapt systems to handle culture-dependent representations and functionality. In practice, the four requirements (Cultural dependencies) described in clause 6 can be divided into two categories: the first relates to the scripts required to present natural language; the second relates to culture-dependent items such as cultural conventions. Subclause 6.2 of this Technical Report is concerned with scripts, whilst culture-dependent items are considered in 6.3.

Note that the installation of locally specific requirements (described in 6.2 to 6.4) onto an INTERNATIONALIZED system ensures the desired behaviour of the localized system. An INTERNATIONALIZED system should, therefore, be culturally neutral for those items described in 6.2 to 6.4.

NOTE 7 This is only a list of the differences or requirements; it is not always necessary to support all these items by means of Internationalization. To try to accommodate all (or any) requirements to make systems universally friendly to all (or any) users inevitably leads to diversification, which is somewhat contradictory to the standardization that both ISO and IEC are aiming to achieve.

6.2 Script (from the process viewpoint)

At the present time, more than 3000 languages are spoken throughout the world. Just over a hundred or so of these languages are actually written. About one half of the world's population uses some version of the Latin script whilst the other half use various different scripts. There are many published methods of expressing these scripts from linguistic points of view.

Present-day computer systems need adaptation to accommodate the scripts discussed below for data entry, processing and information display. Internationalized systems must be capable of supporting all operations in many languages. For this purpose, the script may be seen from the processing viewpoint.

In addition to the linguistic classification, scripts in the world can be categorized into three writing schemes from the processing viewpoint, namely alphabetic (possibly including diacritical elements), syllabic and ideographic.

These different classes of scripts all have unique representation schemes, and implementations of each are separately available. However, each of these scripts contains at least one characteristic which is challenging to implement and these are listed below.

The categorization may vary by the coding method of a coded character set even for the same script, and one script does not necessarily always belong to one category. For example, Korean Hangul is categorized as syllabic in most cases, due to the fact that Hangul is understood as being composed by using Hangul Jamo, this is the linguistic view. If the coded character set includes only Jamo and the combination method of Jamo is required, then it is undoubtedly a syllabic script. On the other hand, if the coded character set includes all possible combinations of Jamo (like KS C 5700), then there is no need to categorize Hangul as syllabic from the processing viewpoint, and it is effectively an alphabetic script. If the coded character set includes only frequently used Hangul (like KS C 5601), then the Hangul can be categorized as an ideographic script due to the open-ended nature of the coded character set (again only from processing view point). Classification is clearly, therefore, dependent upon the coding method used.

These different characteristics significantly increase the difficulty of building one INTERNATIONALIZED system that can run applications in any script, and it is obvious that the complexity of an internationalized system is highly dependent on coded character sets. For simplicity of data processing, therefore, it is highly recommended that the coded character set should be as close as possible to the alphabetic script (even if the linguistic meaning of the writing system is different).

Furthermore, the new multi-lingual and multi-script cultures of the world indicate that specific scripts are no longer systematically and uniquely associated with given linguistic families. Therefore, defining a culture and a specific language no longer automatically imply one script.

6.2.1 Alphabetic scripts

An alphabetic script is a script with (almost) independent symbols which have no specific meaning attached to each symbol (vowels and consonants). It requires a string of such symbols (alphabetic characters) to represent meanings. In most cases each character represents a pronunciation sound, but this is not always true.

About one-half of the world's population uses some version of the Latin script, which is an alphabetic script. In addition, Cyrillic, Greek, Arabic, Hebrew and Japanese Katakana/Hiragana are examples of non-Latin alphabetic scripts from a purely processing viewpoint.

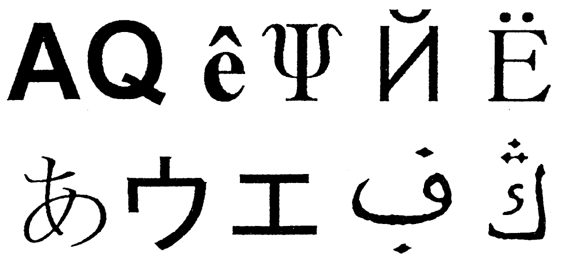


Figure 8 — Examples of Alphabetic characters

Normally, each coded symbol does not have any priority of importance. Vowels and consonants normally have equal importance in alphabetic scripts, and vowels are distributed throughout the alphabets. One of the important features of alphabetic scripts is that the number of characters in each script is relatively stable. This helps the open-ended number possibility of combining characters of the alphabet as required. Most alphabetic scripts also possess both uppercase (such as CAPITAL) and lowercase (such as small) forms of each letter.

In addition, some alphabetic scripts contain diacritical marks or ligatures. Support of such scripts may require great complexity in the technology. However, to facilitate the development of information technology, linguistic categorisations need to become more sensitive to both the limitation and the strengths of the supporting technology of the diacritical marks and/or ligatures. Even so, in the case of alphabetic scripts, the independent coded symbols are a base unit for processing.

6.2.2 Syllabic scripts

In syllabic scripts, vowels are represented either as single characters or as components of characters. If vowels appear above, below, within or beside their associated consonants, the characters maintain individual integrity. Alternatively, vowels and consonants may combine to form single characters, different from either the constituent vowel or consonant. Most South-East Asian scripts contain independent vowels, and Korean Hangul contains vowel consonant character combinations.

In some of these scripts however, vowels are never considered to be separate characters, which illustrates the different definitions of "character" in different cultures. Thus, the processing unit can be either syllabic (different from script to script) or individual vowels and consonants. The number of character components in syllabic scripts is relatively small and stable; on the other hand the number of syllables is far larger, but also stable (not fully open ended).

เจ็ด 칠 सात

Figure 9 — Examples of Syllabic characters

6.2.3 Ideographic Scripts

Ideographic characters, such as Chinese Han characters, symbolise a concept and a sound, or several sounds. Moreover, from the process viewpoint, ideographic scripts have an open-ended nature in terms of the number of characters within one script; moreover, the large number requires a new encoding scheme that invites new processing as a reality.

Apart from their open-ended nature, ideographic scripts are not much different from alphabetic scripts, although the characters look complex in shape and large in number.

Alphabetic and syllabic scripts are all phonetic, with no specific meaning attached to the individual character, while ideographic scripts are normally non-phonetic and has meaning associated to the individual character.

国際化 **規格**

Figure 10 — Examples of Ideographic characters

6.2.4 Script and culture

A specific script is not uniquely associated with a linguistic family. For example, Persian is an Indo-European language, written with Arabic characters which were designed for a Semitic language.

6.3 Culture-dependent items and cultural differences

This subclause lists many culture-dependent items (features) that occur in the processing and presentation of information, and gives examples of some of the different conventions that apply to each feature. The correct handling of each of these conventions can be considered as a requirement for internationalized information processing systems.

The acceptability of various methods of handling these cultural differences is dependent on the local culture. For example, tolerance of inconvenient methods of character input varies with the frequency of the inconvenient characters, with the result that U.S. users may accept the inconvenience of multiple key strokes to enter accented characters more readily than French users, because of the low frequency of accents in English text.

Furthermore, the degree of requirement for some culture-dependent features may be different from one culture to another. For example, the requirement for box-ruling (keisen) in text is very high in Japan, but is relatively low in the U.S. This Technical Report includes such culture-dependent features.

6.3.1 Character encoding and handling

Coded character sets are used for application data, and also for literal, source code, search functions, and identifiers in programming languages etc., and vary according to language and culture. These variations occur in the choice of the characters themselves (for example, national characters), the size (number of bits) of the encoded form of a character (for example, multi-octet coding), and the allocation of characters within the code-table. For example, in different coded character sets the same character may be allocated to different code-table positions corresponding to code elements of the same or different size. Therefore, the character range should not be hard coded.

6.3.2 Writing direction

The direction of writing is dependent on language and culture (e.g. from left-to right, right-to-left, top-to-bottom, etc.). Furthermore, some languages such as Hebrew or Arabic are written from right to left while numbers within the text of these languages are written from left-to-right.

Differences in writing direction can also affect the use of mirrored characters, such as parentheses. In Arabic an "open parenthesis" character is a right parenthesis, but in English an "open parenthesis" character is a left parenthesis. Thus, for parentheses, the terms "open" and "close" are more appropriate for an internationalized system than the terms "right" and "left" (see Annex C).

6.3.3 Character ordering sequence

When character strings are listed in so-called "alphabetic order", or some other conventional order that is familiar to users, the order of the strings depends on the language. For example the German sharp-S is regarded as SS, and Spanish CH appears after CZ.

6.3.4 Multiple forms of characters

Some coded character sets include multiple forms of individual characters, and mapping charts are required to relate the different forms of a character. In some cultures, however, this is unnecessary because each character has only one form. Some examples of such relationships are:

— Normalized-Character,

— Uppercase/Lowercase,

— Free-Standing/Initial-Form/Medial-Form/Final-Form,

— Subscript/Superscript,

— Simplified-Form/Variation-of-Form/Traditional-Form(CJK), etc.

6.3.5 Hyphenation, Spacing and Punctuation

Languages such as English contain hyphens, spaces, and abundant punctuation, while others such as Japanese do not. Thus, hyphenation, spacing, and punctuation differ according to natural language. In addition, the rules for using hyphenation, spacing and punctuation differ according to the language.

6.3.6 Expression of numbers as words

When numbers are expressed as words for the purpose of explanation, numbers with identical numeric format may have different word representations within a language. For example:

— 10 can be written as "ten",

— 1200 can be written as "one thousand two hundred" or as "twelve hundred".

— 10000 is written in Japanese as "man" (single word) and 100000000 as "oku" (a single word).

6.3.7 Character classification

The classification of characters as numeric, alphabetic, etc. differs by culture. Examples are the special characters in the Latin alphabetic coded character sets, and Hanja and Hangul characters in the Korean coded character set might be classified as separated in future.

6.3.8 Messages and Dialogues

If native languages are used for computer-human dialogue, the headings, prompts, error messages, and warnings will be replaced by equivalent text in the local language. The order of words in the local language may need to be different from the original order depending on the grammar of the language. For example the equivalent of "open file" in some languages has the same word order as "file open".

The following considerations should, therefore, be adhered to for all messages and dialogues, including commands, key words and user responses:

— Message text must not be dynamically constructed

— Messages and dialogues must not be stored in a fixed length location

6.3.9 Text length

Equivalent text in different languages may occupy different amounts of storage space in information processing systems.

6.3.10 Spelling

The spelling of some words varies from one culture to another. For example, Center or Centre, Color or Colour, are respectively the U.S. and the U.K. spellings of the same words.

6.3.11 Documentation

Documentation of information processing systems and applications should be provided in the user's natural language. Also, the format of a document such as indentation is cultural dependent.

6.3.12 Layout of text on a page

The rules for the layout of text on a page are culture-dependent. For example the rule that a word at the end of a sentence should not be broken for continuation on the next line.

Also, special page layouts for documents (mainly legal documents) are required in some cultures, such as the centre-folded, double-sided (Fukurotoji) in Japan.

6.3.13 Character (Glyph) size, Line size and Line spacing

The size of printed/displayed characters, the line length, and the spacing between lines, differ among cultures and scripts (for example, the Han script is normally wider than the Latin script).

6.3.14 Font requirements

The preferred style of fonts for rendering of characters differs among cultures even for the same character. For example, Chinese users prefer a more brush-written style to the plainer style that Japanese users like. An unfamiliar font style may give readers a strong foreign impression, so the choice of font may differ between localized applications.

For users in some cultures, the attributes of a character, such as an overline or underline on a character or the style of its background (mesh for example), belong to that individual character (as it's attributes). However users in other cultures may see the overline, underline, or mesh as an attribute of the word or section of text as a whole, independent of the individual characters. In printed form these views do not differ much, however they affect the actions needed at the user-interface to create the text or select the font. This difference may depend on the use either of a character repertoire having a relatively small fixed size, or of a large and open ended character repertoire.

6.3.15 Drawing of boxes in text

The degree of importance of ruled lines (or boxes) in text is different from one culture to another. In fact, one of the largest problems of localized foreign word processors in the Japanese market is their poor capability for drawing boxes.

6.3.16 Icons and Symbols

Icons and standard recognisable symbols are different depending on the country and culture. For example, the icon for a trash can in a user-interface intended for the U.S. is not recognised to be a trash can by Japanese users.

6.3.17 Mathematical symbols

Common mathematical symbols/format are different in some cultures. For example, a dot above and below a horizontal bar is a division symbol for most cultures, but it is a subtraction sign in Denmark, while the position of the minus sign, equation order, etc. are different in Arabic.

6.3.18 Paper sizes

The standard sizes of paper sheets that are in common use for printing in offices (in information processing application) depend on the culture. For example, North American standard letter size is different from ISO standard A4 size.

6.3.19 Data input method

The method of data input differs from one culture to another, and not only because of differences in the layout of character keys on the keyboard. There may not even be a one-to-one correspondence between keystroke and character for some scripts.

Furthermore, in some languages, two or more methods are available for entering characters on the keyboard. For example, on Japanese keyboards the user can enter the characters in Kana and the processor can automatically convert them to Kanji characters. The method in use is selected according to user preference, and the preferred input methods differ from culture to culture.

6.3.20 Voice messages

Some applications require translation of the spoken component of the message into the user's language (but a music component might remain unchanged).

6.3.21 Date, Time and Calendar

The presentation of date, time, and calendar details is culture-dependent; for example, the sequence of writing the day, month, and year. Different presentations of this data are commonly used within a single culture; for example, 09/18/90 and September 18, 1990, or 2:00 p.m. and 14:00. Some countries express year names using their local Eras, while some cultures use a lunar calendar.

6.3.22 Number format

The presentation of numbers is culture-dependent; for example:

- 99,999.99 in some cultures is equivalent to 99.999,99 in others.
- the digit separator character can be the comma (,) , period (.) or space( )
- the digit separator occurs with each 3 digits in some cultures, and with each 4 digits in others.
- the notation of symbolic shapes.

6.3.23 Number rounding

Rules for truncating and rounding numbers are culture-dependent. For example, they can vary depending on the sign of the number, negative or positive.

6.3.24 Currency symbols

Currency symbols can appear at the beginning of a numeric string that represents a currency amount (for example, $15.23 in the U.S.), or in the middle (for example, 15$23 in Portugal) or at the end (for example, 15,23F in France). The currency signs, monetary field size, formatting, etc., are also culture-dependent.

Global currency symbols are defined in ISO 4217.

6.3.25 Price expression

In addition to the different presentation of currency amounts, expressions of price can be different according to the culture (for example, in some cultures $123.45++ means tax and service charge are not included).

6.3.26 Telephone number format

The presentation of telephone numbers varies from country to country. Also the telephone number to be used to call a particular destination telephone varies depending on the location of the caller. For example, 5432-9876 is a number for a Tokyo destination as used by a caller within Tokyo, 03-5432-9876 is a number for a Tokyo destination as used by a caller within Japan, and 81-3-5432-9876 is a number for a Tokyo destination as used by caller from another country.

6.3.27 Postal address format

Presentations of postal addresses vary by country. For example, the state-town-street sequence in China and street town-county sequence in the U.K.. The postal address code uses only numeric characters in some countries, but includes non-numeric characters in other countries.

6.3.28 Measurement system

Measurement systems (for example, distance, weight, speed) are culture dependent. Moreover, some cultures have both a modern measurement system and traditional units. ISO standards define a Global Uniformity measurement system, but this measurement system is not used in daily life in some of countries.

6.3.29 Colour significance

The significance of colours differs depending on the culture; for example, white clothing is used for dead bodies in some Asian cultures. Some languages also have fewer terms for some groups of colours; this usually means that the people using those languages are less sensitive to the differences in colour among those groups of colours.

ISO 3864 is one of the international standards related with the colour significance.

6.3.30 Function names

Sometimes the words of a natural language are used as function names within computer programs. For example, many English function names are used in non-English speaking cultures. In English, the function names usually carry meanings indicative of their functions, but in a non-English speaking culture, that meaning may be lost. However translation of such words to the local language may give rise to other problems.

6.3.31 Personal titles

Methods of addressing people differ from culture to culture.

6.3.32 Taboo words (terms)

Each culture has its own taboo words and terms which are not problematic in other cultures. For example, the word "interest" as a financial term is not acceptable in some cultures.

6.3.33 Regulatory requirements regarding information systems

Many countries have their own regulatory or legal rules for information systems, such as requiring all manuals to be printed in that country's natural language. However, these regulations differ from country to country.

6.4 Additional culture-dependent items

Culture-dependent features in information processing are not limited to those listed in 6.3. As the computer becomes used for a wider range of applications, the number of such features will grow. To satisfy the requirements of all those culture-dependent features by means of one method for the design of systems is not practical or possible. Methodologies to satisfy the INTERNATIONAL requirement need to be flexible enough in principle to allow new local requirements to be added in future.

7 Model and Services required for INTERNATIONALIZATION

This clause introduces the model, functionality, and services required to produce INTERNATIONALIZED/ LOCALIZABLE systems which will satisfy the original requirements described in clause 6. In particular, this clause highlights the functionality and services which programmers expect in programming languages or operating systems (or platforms).

It is expected that most programming languages will provide these services through the particular syntax of each language, or through access to platform-provided services. Each of the services should behave identically toward every different cultural convention that it supports. For example, it is anticipated that programming language services will be able to format numeric values in a manner satisfactory to users in each of the supported cultural environments.

An extension is proposed to the data model for textual data, as typically implemented today. The extension will accommodate character repertoires whose coding methods cannot be handled by the "single-byte equals character" model at present in common use.

Programming languages will use the services described here to facilitate communication between the user and the computer in the user's native language. For example, the native language should be used in computer-generated messages, source code literals, and embedded program comments. Computer-human communication can also be facilitated by a wide range of methods for the user to input identified characters

The diversity of cultures to be supported justifies an implementation strategy which minimises the number of different versions of an application program required to satisfy different cultural environments. For example, current internationalization standards recommend the capability for dynamic selection of cultural conventions at run-time of an application program, and the support of multiple coded character sets. This clause also refers to some standards such as the Universal Coded Character Set (ISO/IEC 10646-1) and POSIX (ISO/IEC 9945), and to industry proposals such as the object-oriented internationalization specifications from industry consortia. These examples should be treated as a base for further discussion and not as an endorsement or a mandatory requirement to support the described services.

7.1 A Model for Information Processing Systems

Before defining the functionality and services needed to meet the original (first level) requirement, the basic assumptions about the system structure must be stated. The assumed structure is called the "model" in this clause. All functions and services described in the following subclauses are based on the model. If a different model were adopted, then the corresponding functionality and services might be different from those described in this Technical Report.

The basic model has the following features:-

a) Application programs are supported by an application platform.

b) The user communicates with the application program through the application platform.

c) The application platform communicates with the user through its presentation layer, and the user communicates with the application platform through its input layer.

d) The character input part of the input layer provides a stream of coded characters to the application platform; at this boundary there is no culture-dependency within the coded character stream. Almost all of the culture-dependent input conversions are handled within the input layer.

e) The output stream of coded characters, which has little culture-dependency, is rendered to culture-dependent shapes within the presentation layer of the application platform. Only a very limited set of culture-dependent processes are handled within the application platform, other than in the presentation layer.

f) In the same way, the culture-dependent processing of a data stream for a communication link (if any) is handled within the communication layer of the application platform.

g) Frequently-used functions and reference data within a repository are placed within the application platform, and those functions and data are shared and used (or referred to) by all parts of the system including the application program. For example, a "clock" for time and date is a part of the application platform, and all operations related to time and date will access this clock to obtain any necessary data. Also, those data should be converted into culture-dependent formats for the user according to the stored reference data within a repository in the application platform. Another example is the reference data within a repository or upper/lower case conversion for the supported culture(s). These data are stored within the application platform for reference when required.

h) On the other hand, rarely-used culture-dependent functions and reference data are placed within those application programs which need to use the rarely-used case data frequently. It is not economical to hold all culture-dependent functions and reference data within application platforms. Postal address format is an example of a culture-dependent feature that would typically be supported by functions and data within an application program.

i) Some of the reference data will provide for all kinds of cultural conventions relevant to a culture-dependent feature; the choice of convention will be selectable by a switch. For other features the reference data can be swapped (from archive storage) to permit a change of convention from one culture to another.

j) LOCALIZATION tools are a particular kind of application program intended to support the LOCALIZATION of the INTERNATIONALIZED application platform and application programs.

The above assumptions make it clear that some culture-dependent functions will be present within every part of a system. It is not possible to consolidate all culture-dependent functions in one place, such as within the application platform. However, to ensure the portability of application programs between different types of application platform, it is important to have a clear definition of the services which the application platform provides for any application program (whether already known, or unknown). It is also important to have a clear definition of all culture dependent functions that an application program offers to its users, and a statement of which functions are assumed to be supported by the application platform and which are provided by the application program itself.

In conclusion, ISO/IEC JTC 1 has extended the model of INTERNATIONALIZATION given in the TSG-1 final report to align it with the model described here. The revised diagram "Model" is shown in FIGURE 11. The model defines a number of service interfaces between its various components. Those can be identified as;

— Application Program Interface (API): This is the interface between the Application and the Application Platform which provides the INTERNATIONALIZATION services. It is the only interface which needs to be visible to an application

— Input/Output (I/O): This is the interface between the I/O components of the Application Platform and the actual I/O devices such as display units, printers, files, communication services etc..

— User (U): This interface can be considered to have two parts - the user interaction through the Application Platform and user interaction with the Application Platform directly. The latter is the service interface referred to within the model and is needed to support direct user control at run-time over cultural conventions, language selection, etc.

— Internal (INT): There are a number of services which are needed within the Application Platform in order to support the interfaces described above; these are identified as internal interfaces.

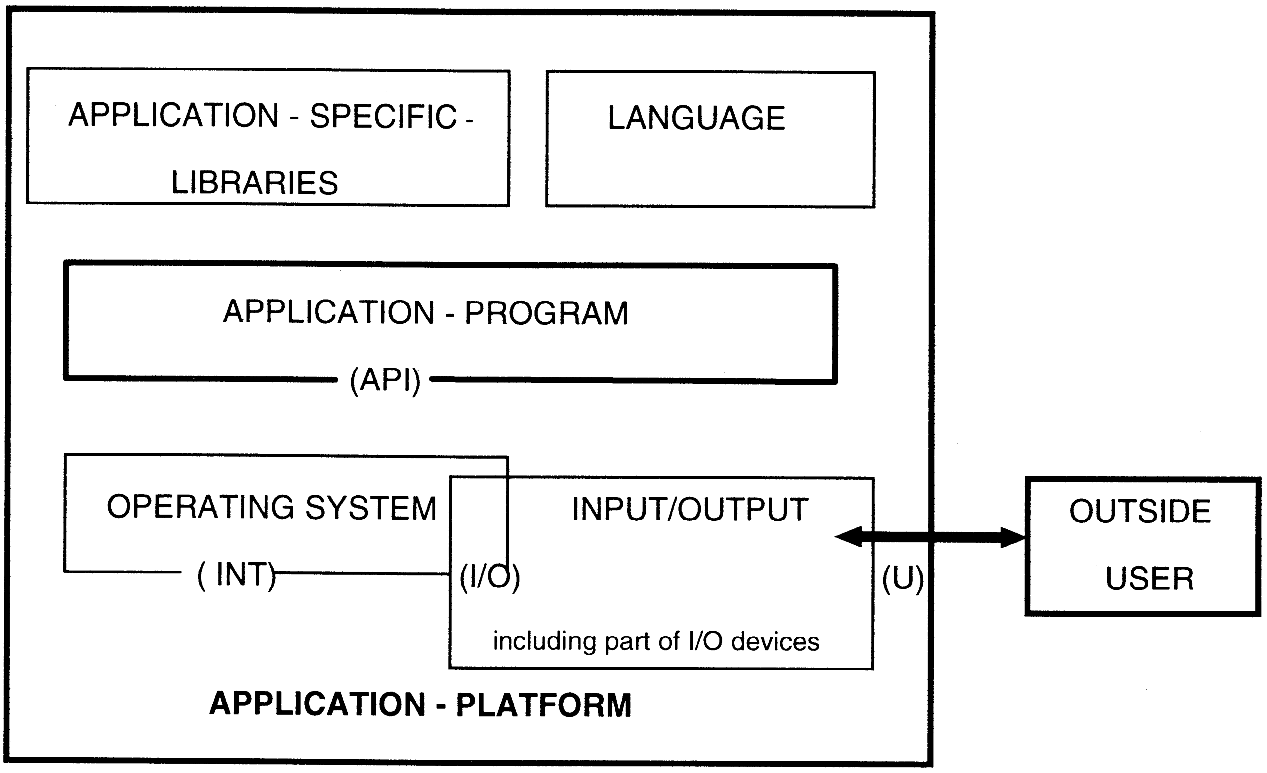
TABLE 2 indicates which of the different services (in 7.2 and 7.3) can be satisfied by each of the service interfaces described above.

7.2 Service Requirements

The service requirements for INTERNATIONALIZATION are identified in this subclause. The services defined in this subclause may, or may not, be part of the platform (or system) depending on the nature of each service. In this context, programming language compilers or interpreters can be considered to be applications running on a platform (hardware plus software). The services can be provided by the platform (such as through POSIX interfaces for INTERNATIONALIZATION to the operating system) or they can be provided as part of the application.

Clearly a solution where these services are provided by the platform for all applications is preferable. Examples of standards which are related to the required INTERNATIONALIZATION services are identified.

Services defined here are intended to satisfy the original requirements described in clause 6, and are suitable for installation into systems. Each requirement is not necessarily satisfied by one corresponding service. For most of the requirements several different services are needed to perform the functions necessary to satisfy them.



NOTE This is an extension of the model in the TSG-1 final report.

Figure 11 — Model

7.2.1 Coded Character Set and Data Representation Service

7.2.1.1 User Requirements

Dialogues in the local language between users and system platforms or applications require the support of language-specific coded character sets. For example, German text contains "umlauts" and the "sharp S" - these are characters which are not used in the English language, but are essential in German.

Most languages have requirements either for extra characters which are not included in the ISO/IEC 646-IRV set (often called ASCII, American Standard Code for Information Interchange), or for an entirely different character set to support a different alphabet or script. The coded character set comprises characters which meet the needs of one or more scripts. The scripts are classified into various groups based on their fundamental characteristics as described in 6.2. Also the characters have to be processed correctly in accordance with the conventions of the target culture.

Note that the composite sequence may need to be considered as the equivalent of a character.

7.2.1.2 Coded Character Set Handling Service

To support the user's language(s) in the user's script(s), it is necessary to support one or more coded character sets which include the user's script(s). Thus support of the necessary coded character sets must be provided throughout the system. The Coded Character Set Handling Service provides the capability to recognise, process, store, retrieve, communicate, and present different coded character sets.

The following significant attributes of coded character sets should be noted, since they may differ between one set and another:

— the number and size of bytes used for coding each character of the set (the coding method),

— the type of combining method allowed for creating composite characters, if required,

— the behaviour of the scripts, as described in 6.2.

Table 2 — Categories of INTERNATIONALIZATION services (in 7.2)

Application Program Interface (API)	
Coded Character Set Handling Service	7.2.1.2 (part)
Coded Character Set Invocation Service	7.2.1.3 (part)
Message Service	7.2.1.8.4 (part)
Compare, Sort and Search Service	7.2.2.2
Case Mapping Service	7.2.2.3
Date Format Service	7.2.4.1
Time Format Service	7.2.4.2
Day Numbering Service	7.2.4.3
Week Numbering Service	7.2.4.4
Numeric Formatting Service	7.2.4.5
Currency Formatting Service	7.2.4.6
Cultural Convention Synchronisation Service	7.3.4

continued

Table 2 (continued)

Internal Services (INT)	
Coded Character Set Handling Service	7.2.1.2 (part)
Coded Character Set Invocation Service	7.2.1.3 (part)
Coded Character Set Identification Service	7.2.1.4
Coded Character Set Registry and Repository Service	7.2.1.5
Data Announcement Service	7.2.1.6
Coded Character Set Identification	7.2.1.7
Data Class Definition Service	7.2.2.1
Data Conversion Service	7.2.2.4
Multi-Lingual Synchronisation Service	7.2.3.2 (part)
FDCC-set Repository Service	7.3.2.1
FDCC-set Invocation Service	7.3.2.2
Input/Output Services (I/O)	
Coded Character Set Handling Service	7.2.1.2 (part)
Data Presentation Service	7.2.1.8.1
Data Input Service	7.2.1.8.2
Data Communication Service	7.2.1.8.3
Message Service	7.2.1.8.4 (part)
User Services (U)	
Coded Character Set Invocation Service	7.2.1.3 (part)
Language Selection Service	7.2.3.1
Multi-Lingual Synchronisation Service	7.2.3.2 (part)

Examples of the various coding methods used in different standards for coded character sets are:

1) single-byte coding with 7-bit bytes (such as ISO/IEC 646),

2) single-byte coding with 8-bit bytes (such as ISO/IEC 8859-1),

3) multiple-octet-byte single-byte coding (such as ISO/IEC 10646-1),

4) single-octet-byte multiple-byte coding (such as JIS X 0208 which uses two 7-bit bytes per character. or UTF-8 form of ISO/IEC 10646),

5) multiple-octet-byte multiple-byte coding (such as UTF-16 form of ISO/IEC 10646).

The Coded Character Set Handling Service ideally should provide a service for all of these types of coded character sets.

Most of the coded character sets (such as ISO/IEC 646) do not include any "non-spacing" characters. If necessary, characters may be combined by the use of control functions, for example, to combine a diacritical mark with a base letter. However ISO/IEC 10646-1 has combining characters which may be used to create "composite sequences" for various purposes.

Association of a single coded diacritical mark with a separately coded base letter is a primary feature of ISO 6937. On the other hand, ISO 8859-1 does not allow such techniques of character composition. When composite sequences, or combining characters, are used, the equivalence and non-equivalence of such characters with coded pre-composed characters should be considered during processing of character strings.

Some standards related to coded character set are:

- ISO/IEC 646
- ISO 6937
- ISO/IEC 8859-1 (and other parts of ISO/IEC 8859),
- ISO/IEC 10646-1
- JIS X 0201
- JIS X 0208
- JIS X 0212

7.2.1.3 Coded Character Set Invocation Service

Since different coded character sets are used to present different languages, a Coded Character Set Invocation Service is necessary. The Coded Character Set Invocation Service provides the capability to specify the coded character set to be used for input, processing, and output of data. This functionality can be invoked in any of the following ways:

- user selection
- default specification
- data announcement techniques
- information about the presentation capabilities of specific output devices

The service may also allow the user to switch dynamically from one coded character set to another, if required.

Some standards related to this service are:

- ISO/IEC 2022
- ISO/IEC 9945-1

7.2.1.4 Coded Character Set Identification Service

When different coded character sets are used to represent text in different languages, and those sets are switched according to the requirement, then a method is needed to identify the coded character set with a coded identifier (such as a character string) that can be stored or communicated as an item of data.

The Coded Character Set Identification Service provides unique identification of coded character sets. This service allows different coded character sets to be used concurrently on a system or in an application without the risk of data corruption. It also provides identification during the exchange of data between systems or networks, and allows the selection of appropriate translation tables between different encoding of the same character repertoire.

Some standards related to this service are:

- ISO/IEC 2022
- ISO 2375
- ISO/IEC 7350
- ISO/IEC 8824
- ISO/IEC 8825

7.2.1.5 Coded Character Set Repository and Registration Service

The Coded Character Set Repository Service provides a central repository that contains coded character sets and relevant information about them. It is used by services related to coded character sets and data representation. Entries in the repository may include:

- Coding method (7.2.1.2)
- Character-path direction (6.3.2)
- Coded character set identifier (7.2.1.4)
- Character classes (6.3.7)
- Mapping rules (6.3.4)
- Code extension techniques (7.2.1.2)

The requirement for the repository of coded character sets implies that such sets need to be registered. Registration requires some methods of specification for the registered sets, and a registration authority.

Some standards related to this service are:

— ISO/IEC 2022

— ISO 2375

— ISO 7350

7.2.1.6 Data Announcement Service

The Data Announcement Service provides the capability to recognise the coded character set used by data entities (files, messages, etc.). This capability allows the processing and storage of data using different coding methods within the same system, but avoids the risk of data corruption. International Standardization bodies are presently addressing the announcement mechanism.

7.2.1.7 Identification of individual coded characters

A large amount of character-related data must be defined to provide the services described in the previous subclauses. This data will need to include a unique identification for each character to which the data is related. For many purposes it will be an advantage if these unique identifications are short forms which can be stored as compact items of data.

From this point of view, the unique character names defined in ISO/IEC 10646-1 are too long for some of purposes. A short unique identifier for each character should be specified as an alternative. A short mnemonic name for each character would be ideal in principle. But due to the fact that the total number of characters is so large, and that the assignment of easily memorable names is culture-dependent, it is not practical to look for universal short mnemonic names for all of the characters. Instead the assignment of mnemonic names should be included during the localization of the part of the user-interface that is related to the input of character data.

ISO/IEC JTC 1 is in process of development of the short identifier of the characters as amendment-9 of ISO/IEC 10646-1: 1993.

7.2.1.8 Other Character related Services

The usefulness of the various coded character set related services will be considerably weakened without the availability of certain associated input/output and message services, as detailed in this subclause.

7.2.1.8.1 Data Presentation Service

The Data Presentation Service provides the capability to present data on different display units, printers, or other output devices. Using the rules stored in a repository, this service includes escapement of characters and selection of different shapes. Preparing data for presentation may involve extensive translation and/or transliteration due to hardware selections or limitations. The service also provides default presentation forms for coded characters that have no associated graphic shape. It is recommended that this service should include the functionality to handle control of the presentation of bi-directional text and some kinds of vertical writing, as well as presentation forms, in order that the character code within the software can be free from culturally dependent presentation style requirements.

Data presentation includes several components such as a font selection service, for which the requirements vary widely from application to application. Thus, it is better to handle each component of the presentation service as an independent application-oriented service.

The basic and common parts of the presentation service are mostly related to hardware.

A standard related to this service is:

— ISO/IEC 6429

7.2.1.8.2 Data Input Service

Having standard interfaces for a wide range of input methods for a wide range of characters is a part of INTERNATIONALIZATION. At such an interface, different characters should simply be different character code values, which are to be processed uniformly. Complex culture-oriented input methods and keyboards are matter of LOCALIZATION.

Some standards related to this service are:

— ISO 1092

— ISO 2126

— ISO 8884

— ISO/IEC 9995

— ISO/IEC CD 14755

— CSA Z243.200

7.2.1.8.3 Data Communication Service

The Data Communication Service provides the capability to transmit and receive data to and from communication systems, while maintaining the integrity of the data. For this purpose, the services described in 7.2.1 (such as the data identification service) should be used for both internal and communication purposes. In international communication environments, this may include data translation due to different coded character sets being used in different service categories.

7.2.1.8.4 Message Service

The Message Service provides the capability to present (display, print, etc.) messages, menus, forms, help information, and on-line documentation in the language selected by the user. Different languages can be used simultaneously. The service maintains the independence of the messages from the applications, allows for variable message length (German translations of English messages tend to be 30% longer), has a delivery service to insert parameters into translated messages using the correct grammar and syntax, and uses the cultural convention repository for format definitions. It also allows users to interact with applications and operating systems in the language of their choice. It allows the entering of local language, using local characters, and parsing of local formats as defined in the cultural conventions repository.

A well defined message service is at the heart of INTERNATIONALIZATION functionality, and substantially reduces the resources required for LOCALIZATION.

7.2.2 Services for character data processing

The services described in 7.2.1 are for handling characters as they stand. It is a common requirement for characters to be processed as a part of data in most data processing applications; additional services are needed in order to process characters in culturally correct ways.

7.2.2.1 Data Class Definition Service

Characters have different character classes. Since processing often depends on the classification of characters as space, numeric, alphabetic, or as special characters, a service is provided to identify these classes.

7.2.2.2 Compare, Sort, and Search Service

Comparison, sorting and searching of characters are culturally dependent. Since these functions occur in many places within information systems, there are two kinds of requirements to be satisfied. The first is that the Compare/Sort/Search result should be fully predictable, and the second is the need for a standard method of access to the service.

Annex D contains a detailed example of this type of service.

A standard related to this service is:

— CSA Z243.4.1

7.2.2.3 Case Mapping Service

Some scripts have upper case and lower case characters. The Case Mapping Service provides upper case to lower case and lower case to upper case mapping.

This idea can be extended to character conversion mapping services which support more than conventional case conversion, for example conversion between traditional characters and modern characters in the ideographic script.

7.2.2.4 Data Conversion Service

The same language can be presented using several different coded character sets, for example, ISO/IEC 10646-1 includes all characters in ISO 8859-1. Thus Danish can be written in both ISO/IEC 10646-1 and ISO 8859-1. The data conversion service has to provide a conversion service between two such coded character sets whenever it is necessary. A transliteration service might be another form of data conversion service.

A standard related to this service is:

— ISO 6936

7.2.3 Services for multi-lingual support

Mono-lingual support for variety of cultures can be performed by supporting and processing selected coded character set(s) in the correct way. Multi-lingual support, however, requires additional services which are unique to the multi-lingual case.

If services are required for multi-lingual support, then there will be substantial differences between a mono(or bi)-lingual system and multi-lingual system. This is contrary to the principal goal of INTERNATIONALIZATION. Services for multi-lingual support, therefore, should to be minimised and should be as independent from any other services as possible.

7.2.3.1 Language Selection Service

This service provides the capability for the user to specify the language of interaction with the application. If none is chosen, the default language is selected.

Note that this service may not necessarily be the same as the FDCC-set invocation service described in 7.3.2.2. The wider the view that is taken of internationalization, the less relation there will be between language selection and the determination of culture from the FDCC-set.

7.2.3.2 Multi-Lingual Synchronisation Services

The Multi-Lingual Synchronisation Service provides the capability to support more than one natural language simultaneously. For example, a word processor might be required to work with text in Japanese and French on the side by side pages, with synchronised paragraphs.

7.2.4 Services for Common Cultural Conventions

For maximum user friendliness, each cultural convention must be appropriate to the user's culture. There is a group of cultural conventions which most application programs will need to use. These cultural conventions can normally be used by any application. The services required for this group of cultural conventions, other than character-related services, are described in the following sub-clauses.

7.2.4.1 Date Format Service

Normally, a computer system holds one item of data for the date which may be used to control all date related functions throughout the system. The date format required for reading by humans, however, varies from culture to culture as described in 6.3.21.

The Date Format Service provides the capability for the user to select the appropriate format.

Some standards related to this services are:

— ISO 8601

— JIS X 0301

7.2.4.2 Time Format Service

Time format requirements are different from culture to culture. However, the computer system only requires one time data throughout a single system. The Time Format Service has the capability to handle the various formats as well as taking account of world time zones and their offset values relative to UTC.

Some standards related to this service are:

— ISO 8601

— JIS X 0302

7.2.4.3 Day Numbering Service

In some countries a week begins on a Monday, in others on a Sunday, and it begins on a Saturday in some Islamic countries. The day numbering service provides the number of the day of the week. Note that, in addition, the names of the days of the week need to be represented in the user's language.

A standard related to this service is:

— ISO 8601

7.2.4.4 Week Numbering Service

In some applications it is often more convenient to use week numbers for calculations than months and days. The first week in a year is defined differently in various countries. The Week Numbering Service supports these conventions and provides conversion routines.

A standard related to this service is:

— ISO 8601

7.2.4.5 Numeric Formatting Service

The interpretation of numeric fields in unfamiliar formats is one of the major contributors to human errors in data processing. Within a computer system, one single method is used to express a number for machine processing purpose; however this is not friendly to most of users. The Numeric Formatting Service provides the capability to handle different cultural conventions, such as the point decimal delimiter used commonly in the U.S. and the U.K. vs. the comma used in most of Europe. Other differences are the use of spaces or periods to separate groups of 3 digits, and the identification of negative numbers by use of leading or trailing minus signs or by surrounding parentheses.

A standard related to this service is:

— ISO 6093

7.2.4.6 Currency Formatting Service

The Currency Formatting Service describes the handling of currency fields and symbols. The symbols for currencies vary from country to country, and their placement before, after, or between the integer and the fractional part of the amount varies as well. The field lengths and the number of digits after the decimal point depend on the monetary system. Negative amounts are indicated according to local rules and regulations.

A standard related to this service is:

— ISO 4217

7.2.5 Services for other cultural conventions

There is another group of cultural conventions which are very important for some types of application program but less so for others. For example, the postal address format are very important for certain applications, but are of no significance for most applications.

Support services for those less common cultural conventions are required in a similar manner to those for commonly used cultural conventions.

However, the placement of such a support service within a system may be different from the more commonly used cultural conventions.

A sample of services for the other cultural conventions follow:

7.2.5.1 Measuring System Service

Presentation of dimensions in inches, feet, yards, and miles are different from millimetres, centimetres, meters, and kilometres; ounces and pounds convert into grams and kilograms, cups and gallons into litres, and degrees Fahrenheit into degrees Celsius. Conversion facilities and country specific presentation formats are provided by the Measuring System Service.

A standard related to this service is:

— ISO 1000

7.2.5.2 Paper Size Service

This service provides the capability to select various paper sizes according to cultural requirements.

Some standards related to this service are:

— ISO 216

— ISO/IEC 6429

— ISO 6716

NOTE 8 ISO/IEC 6429 is a standard for control, and includes the control sequences that define common paper sizes in different parts of the world.

7.2.5.3 Postal Address Format

As described in 6.3.28, postal address formats are different from culture to culture. It is very doubtful whether it is possible to provide an automatic format conversion service for all postal addresses. On the other hand, a different type of consideration is necessary for this cultural convention. For example, the postal code in the U.S. (ZIP code) is expressed using only numbers, whereas in the UK, it also includes alphabetic characters. Therefore, for some applications, data validation design needs to consider this cultural difference.

A standard related to this service is:

— ISO 11180

7.2.5.4 Other cultural conventions

There are a number of other cultural conventions listed in 6.3, many of which are not commonly used. Furthermore, new cultural conventions which will need to be recognised in the future, due to the open ended nature of cultural conventions, will be categorised as less commonly used during the early stages of recognition.

However, as technology progresses, and more system resources become economically available for the user, then some of less commonly used cultural conventions will be re-categorised as commonly used cultural conventions in the future. There can be no clean and clear separation between common and less common; there will always be gray areas.

7.3 Organization and Management of cultural conventions

Cultural conventions can, in theory, be handled at any place within a computer system. However, the model presented in 7.1 indicates where it is anticipated that the various services will, in practice, be situated.

In particular, most of the commonly-used cultural conventions are expected to be handled by services situated within the application platform. Less commonly-used cultural conventions, on the other hand, are expected to be handled by services situated within each independent application program. It is possible that some of these may initially be grouped together as utilities on a commercial basis, with the result that de-facto standards will be created by these commercial utilities.

7.3.1 The Formal Definition of Cultural Conventions (FDCC)

Before a cultural convention can be used in the creation of an appropriate cultural environment, there must be a formal definition of the cultural convention for that cultural environment. It is not proposed to discuss this process in any detail in this Technical Report, as this will be subject of subsequent International Standards; however it is appropriate to briefly outline the key stages in this process and the subsequent use of these formal definitions. In this Technical Report the term FDCC will be used as a short-hand for the formal definition of a cultural convention.

The definition of FDCCs for cultural conventions defined as an application platform responsibility must be the same for any application platform in order to ensure portability of applications between application platforms. Standards for the definitions (specification method) of FDCCs will, therefore, required.

Note that the cultural conventions that are to be supported by application platforms may be up-dated from time to time as a result of developments in technology and/or market pressures. It is important that the FDCCs supported by application platforms are clearly and consistently defined, and such definitions should, therefore, always be provided via International Standards.

Note also that with these application platform supported FDCCs, it is possible to provide pre-localized internationalized applications. In many cases, therefore, these cultural conventions might be seen as being the only cultural conventions necessary for the internationalization of information systems. However, without additional cultural conventions which are supported within the application it is not possible to provide the comprehensively internationalized applications which many users require.

7.3.2 The Formal Definition of a Cultural Environment

In order to provide the information necessary for the definition of a cultural environment appropriate to a specific culture it is first necessary to collect the set of FDCCs which define the various cultural conventions appropriate to that culture. Such a set will be referred to in this Report as an FDCC-set.

These FDCC-sets may need to be standardized or registered in an appropriate internationally accessible manner in order to ensure that applications can utilize the various FDCCs in a consistent manner.

7.3.2.1 FDCC-set Repository Service

The FDCC-set Repository Service provides the capability to maintain and access the Formal Definitions of Cultural Conventions (FDCC), and to maintain the collections of such FDCCs which have already been defined as FDCC-sets. The repository, therefore, contains the information that supports the other cultural convention services.

Note that the FDCC-set repository service is only possible for FDCCs that are supported by the application platform, and that, therefore, it is not practical to expect every item of culturally dependent information to be provided by the repository service.

A standard related to this service is:

— ISO/IEC 9945-2

7.3.2.2 FDCC-set Invocation Service

Once more than one FDCC-set is available on an application platform then it is necessary to be able to select the required FDCC-set. The FDCC-set Invocation Service provides the capability to invoke the FDCC-set (by default from the FDCC-set repository) as requested by the user or by the application.

7.3.3 Cultural Conventions within Application Programs

Cultural conventions that are to be supported by application programs and/or by independent library functions are, as already discussed, open ended. It is not practical, therefore, to define and specify all of those cultural conventions in a standard manner. However, if application providers do not have enough information about these cultural conventions, it is not possible to provide internationalized applications utilising these cultural conventions.

One possible solution would be the provision of data books or on-line information sources which itemise these cultural conventions and describe the culturally different requirements about each cultural convention. Such data cannot be perfect, however, and will be the best available knowledge base of the day, requiring periodical up-dating.

7.3.4 Cultural Convention Synchronisation Service

Because cultural conventions need to be supported in several places within computer systems, it will be necessary to provide mechanisms to synchronise FDCCs and FDCC-sets which are handled in different places. The FDCC-set Invocation Service only applies to FDCCs implemented within application platforms, and it is, therefore, necessary to have a synchronisation service to ensure that the modification of the cultural content of the various FDCCs, especially those implemented within applications, are synchronised with each other.

A similar situation arises with FDCCs that are implemented on different platforms that communicate with each other.

7.3.5 Considerations for distributed applications

The current model for producing internationalization was developed at a time when systems were typically standalone, with character based terminals. For such environments, the current global internationalization model based on "setting FDCC-set data for selected culture" is a reasonable solution, but it has its limitations; for example, in distributed software or software designed to interact simultaneously with several users, each of whom uses different customs and conventions.

The simplicity of the current FDCC-set model makes it easy to write internationalized mono-lingual programs. At the same time it makes it difficult to write applications that deal with many different languages simultaneously.

Examples of such programs include:

— A spread sheet program that uses multiple currencies and date formats.

— Ledger programs for international companies.

— Database programs that merge and process data bases from different languages or coded character sets.

— Window manager programs that control and label the windows of all applications displayed on a workstation.

— The portion of a windowing system library that creates the resource data base for an application, merging data bases from several systems and sources, each of which could be created in a different language and coded character set.

— Window based process control applications; that is, a single application that interacts with many different users. Each user wants to interact in a different language and each user expects data to be processed (that is, number formats, time and date, sorted lists, ...) per his or her local rules.

— A word processing application for multi-language texts (such as translated documents, international standards and annotated literature), with the need for language sensitive hyphenation, justification, and so on.

Such multi-lingual applications need to be able to call functions (services) that provide the current internationalization services for data consisting of an arbitrary mixture of languages and coded character sets. Consider two cases of remote procedure calls dealing with international text in a heterogeneous network. The simple case (single thread, single FDCC-set) allows the handling of international text with the current FDCC-set model. The more complicated case (multiple threads, multiple FDCC-sets) illustrates remote procedure call problems that cannot be resolved with the current FDCC-set model.

In a threaded environment, two or more threads may be in execution phase at any given time. If either of these threads affects the default FDCC-set, any FDCC-set sensitive operations in the other thread will be affected:

— The global state cannot be maintained by saving and restoring the FDCC-set without blocking other threads.

— FDCC-set sensitive functions cannot be guaranteed to take place in a given FDCC-set without blocking all other threads.

Given that it were possible to specify a FDCC-set as an additional argument to FDCC-set sensitive functions, it cannot be guaranteed that if a client passes some FDCC-set identifier to a server then the identifier can be used to reconstruct the original FDCC-set. Some type of FDCC-set agreement needs to take place.

For all of the above situations that cannot be handled by the current FDCC-set, a new type of (extended) FDCC-set approach may provide a common solution for these issues.

Further technological development is necessary in order to have a stable solution for this problem.

7.4 LOCALIZATION related services

Once various FDCC-sets, and the related services, are available, it will be possible to provide INTERNATIONALIZED systems which can then be LOCALIZED to conform to specific cultural requirements. However, when considering the goal of INTERNATIONALIZATION, the services described above are not sufficient in themselves to meet the user's expectation. In particular, users will require LOCALIZED system of the same quality, cost and timing for market introduction everywhere in the world, in forms suitable for any necessary cultures. To meet this final goal some additional services are necessary to support LOCALIZATION.

7.4.1 LOCALIZATION Support Services

Once an INTERNATIONALIZED system is available, most of the work for LOCALIZATION will be message translation into the local language in the user's script(s). The translation work should be done in an organised fashion by using LOCALIZATION tools to be defined/developed in the near future. Note that this does not mean machine translation of the language, but rather that management tools will be necessary to ensure consistency of translated terminology and to indicate the messages which need to be translated.

7.4.2 Maintenance Services for the LOCALIZED system

To ensure appropriate delivery schedules and quality of updating and maintenance, various Maintenance Service Tools will also be needed. Once LOCALIZATION (and Customization for specific needs as well) has been carried out on a specific application platform and/or application programs, the resulting products that are delivered to the users will not be identical with the original products. However, the provider of the original INTERNATIONALIZED products has a responsibility for the maintenance and up-dating of the LOCALIZED systems which the original provider has never seen. It will, therefore, be necessary to provide maintenance and updating tools to overcome this "never seen" problem.

7.5 INTERNATIONALIZATION in Fortran - A Possible Approach

The following description indicates one way in which the functionality and services required for support of INTERNATIONALIZATION might provided in Fortran. It is not intended to imply that this is a recommended approach, it is simply presented as an example of an existence idea/proof to indicate that it would not be too difficult to add the necessary functionality to the current version of major programming languages. Similar approaches could be used in any other programming languages which contain suitable language extension facilities.

This example shows one way in which two key issues might be approached, namely the identification and/or specification of an appropriate cultural environment, as specified by a FDCC-set, and the use of that cultural environment to automatically invoke the culturally appropriate form of string comparison to enable an array, or other collection, of textual items to be sorted in the correct order for the environment.

In this approach three new intrinsic procedures provide all the necessary control, and these are briefly described first in a simplified version of the style used in the Fortran Standard (ISO/IEC 1539: 1991).

a) `CULTURAL_ENVIRONMENT (FDCC_SET)`

Description: Returns the processor dependent code for the cultural environment specified.

Class: Inquiry function.

Argument: `FDCC_SET` is optional, but must be scalar and of type default character if present.

Result Type: Default integer.

Result Value: If `FDCC_SET` is present, then it must represent the name of a FDCC-set supported by the processor; the result will be a processor-dependent integer which will identify this a FDCC-set within this processing system. If `FDCC_SET` is not present the result will be the processor-dependent integer which identifies the cultural environment in which the processor is currently operating.

b) `REPERTOIRE_KIND (ENVIRONMENT)`

Description: Returns the kind type of the character repertoire associated with the current cultural environment.

Class: Inquiry function.

Argument: `ENVIRONMENT` is optional, but must be scalar and of type default integer if present.

Result Type: Default integer.

Result Value: If `ENVIRONMENT` is present then it must represent the integer code for a cultural environment specified by an FDCC-set supported by the processor; the result will be the kind type of the character repertoire which is associated by default with this cultural environment. If `ENVIRONMENT` is not present the result will be the kind type of the character repertoire which is associated by default with the cultural environment in which the processor is currently operating.

c) `SET_ENVIRONMENT`
`(ENVIRONMENT,CHAR_KIND)`

Description: Changes the current cultural environment.

Class: Subroutine.

Arguments:

`ENVIRONMENT` must be scalar and of type default integer. It specifies the cultural environment to be used for subsequent processing.

`CHAR_KIND`(optional) must be scalar and of type default integer.

If present, it specifies the kind type to be used by default for any subsequent character declarations;

If absent, the kind type associated by default with the environment specified will be used.

An example of the use of these procedures to automatically localize a program to the environment which is current when the program commences execution might be as follows:

```
PROGRAM Automatic_Localization

IMPLICIT NONE

! Establish current cultural environment

INTEGER,ARAMETER::environment=CULTURAL_ &

                  ENVIRONMENT(), ch_kind &

                  =REPERTOIRE_KIND()


!  Character variable declaration

CHARACTER(KIND=ch_kind,LEN=20):: &

string1,string2

!  etc.

.

.


!  Start of execution

CALL SET_ENVIRONMENT (environment)

.                ! This is not strictly necessary,

.                ! but is Probably good practice

.


END PROGRAM Automatic_Localization
```

One of the effects of setting an environment could be to overload the comparison operators between character strings of the character kind specified or implied so as to use the correct culturally comparison algorithm.

Thus the statement

```
        IF (string1 < string2) THEN

          .

          .

          .

END IF
```

which would normally compare two strings character-by-character using the rules specified in the Fortran Standard would compare them using the correct culturally sensitive algorithm once a `SET_ENVIRONMENT` statement had been obeyed. Annex D contains an example of how this might be achieved in Fortran.

A call to a standard sorting routine would then carry out the sorting using this same algorithm without the need to make any adjustment to the sorting procedure at all.

There are many other INTERNATIONALIZATION functionalities which could be incorporated into Fortran in a similar way with relatively little effort, and with only minimal extensions to the language. It is believed, moreover, that this functionality could initially be added by means of a module, in similar fashion to that demonstrated for varying length string datatype in ISO/IEC IS 1539-2:1994.

Annex A

Bibliography

ISO 216:1975[1], *Writing paper and certain classes of printed matter — Trimmed sizes — A and B series.*

ISO/IEC 646:1991, *Information technology — ISO 7-bit coded character set for information interchange.*

ISO 1000:1992, *SI units and recommendations for the use of their multiples and of certain other units.*

ISO 1092:1974[2], *Adding machines and calculating machines — Numeric section of ten-key keyboards.*

ISO/IEC 1539:1991, *Information technology — Programming languages — FORTRAN.*

ISO/IEC 1539-2:1994, *Information technology — Programming language — FORTRAN part 2: Varying length character strings.*

ISO/IEC 2022:1994, *Information technology — Character code structure and extension techniques.*

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

ISO 3864:1984, *Safety colours and safety signs.*

ISO 4217:1995, *Codes for the representation of currencies and funds.*

ISO 6093:1985, *Information processing — Representation of numerical values in character strings for information interchange.*

ISO/IEC 6429:1992, *Information technology — Control functions for coded character sets.*

ISO 6716:1983, *Graphic technology — Text-books and periodicals — Sizes of untrimmed sheets and trimmed pages.*

ISO 6936:1988, *Information processing — Conversion between the two coded character sets of ISO 646 and ISO 6937-2 and the CCITT international telegraph alphabet No. 2 (ITA 2).*

ISO/IEC 6937:1994, *Information technology — Coded graphic character sets for text communication — Latin alphabet.*

ISO/IEC 7350:1991, *Information technology — Registration of repertoires of graphic characters from ISO/IEC 10367.*

ISO 8601:1988, *Data elements and interchange formats — Information interchange — Representation of dates and times.*

ISO/IEC 8824:1990, *Information technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8825:1990, *Information technology —Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO 8859-1:1987, *Information processing — 8-bit single-byte coded character sets — Part 1: Latin alphabet No. 1.*

ISO/IEC 9945-1:1996, *Information technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].*

ISO/IEC 9945-2:1993, *Information technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities.*

ISO/IEC 9995:1994, *Information technology — Keyboard layouts for text and office systems* (all parts).

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

ISO 11180:1993, *Postal addressing.*

ISO/IEC 14755[3], *Information technology — Input methods to enter characters from the repertoire of ISO/IEC 10646 with a keyboard or other input devices.*

[1] Currently under revision.

[2] Replaced by ISO/IEC 9995:1994, parts 1 to 7.

[3] To be published.

ISO/IEC JTC 1 TSG-1 Final report,

NOTE A1 This is an ISO/IEC JTC 1 internal document, document number: ISO/IEC JTC 1 N1335 - April 1991.

CSA Z243.200:1992, *Canadian Keyboard Standard for the English and French language.*

CSA Z243.4:1987, *7-Bit and 8-Bit Coded Character Sets for Information Processing and Interchange.*

CSA Z243.4.1:1992, *Canadian Alphanumeric Ordering Standard for Character Set of CSA Standard CAN/CSA Z243.4.*

JIS X 0201:1976, *Code for information interchange.*

JIS X 0208:1990, *Code of the Japanese Graphic Character Set for Information Exchange.*

JIS X 0212:1990, *Code of the supplementary Japanese graphic character set for information Interchange.*

JIS X 0301:1977, *Identification Code of Dates.*

JIS X 0302:1977, *Identification Code of Times.*

KS C 5601:1987, *Code for information interchange.*

KS C 5601:1992, *Code for information interchange.*

KS C 5700:1995, (Korean version of ISO/IEC 10646-1:1993).

NOTE A2 CSA: Canadian Standards Association

NOTE A3 JIS: Japanese Industrial Standards

NOTE A4 KS: Korean Standard

Annex B

Samples of cultural conventions to be included in FDCC-sets

This annex lists a sample of cultural conventions which are recommended to be a part of a FDCC-set. Further, it is recommended to list:

a) Cultural conventions which are not in this list and which are included in the FDCC-set of a standard or product should be identified.

b) Cultural conventions which are in this list and which are not included in the FDCC-set of a standard or product should be identified, with a justification for their exclusion.

The recommended cultural conventions for inclusion in FDCC-sets are:

— Calendar

— Case Mapping

— Currency symbols

— Date format

— Number format

— String ordering and Comparison

— Time format

Annex C

Bi-directional text

This annex explains some of the aspects of Arabic and Hebrew languages, which are written in bi-directional form, in order to assist in the understanding of some parts of this Technical Report.

C.1 Writing direction

In many coding systems, the definition of certain control functions assumes that data associated with these functions will be serially processed forwards.

The term "forwards" is generally understood to mean a left-to-right data path; this interpretation corresponds with the normal direction of writing in most languages. However, for Semitic languages, where the normal direction of written texts is from right-to-left, but where numerals are written from left-to-right, the term "forwards" is not adequate. Additional processing must be implemented to ensure that control functions operate correctly to obtain desired results. This additional processing must attempt to develop the highest possible degree of applications transparency in order that a single application may be able to process both left-to-right and right-to-left data paths, or even a mixture of both types of path.

C.2 Contextual Analysis

In some scripts, such as the Semitic scripts, a character may assume several different forms depending on the place of the character within the word, as well as on the characters on either side of it.

For these languages, there is no one-to-one correspondence between a character and that character's graphic form. This graphic form is determined by a process of context analysis. For example, in Arabic, each character may have up to four different graphic forms: initial, medial, final or isolated.

C.3 Space generation characteristics

The term "space generation" designates the movement of the current position after presentation (display or print) of the character.

In Latin-based languages, the current position moves forward from left to right by one character space after a character has been presented. At the end of each line, the current position returns to the beginning of the following line, which is the line immediately below the current line.

In Latin-based languages, characters are written from left-to-right and lines from top-to-bottom.

For some of other languages, this order does not apply. In Arabic and Hebrew, the non-numeric character path is from right-to-left, while numeric characters are represented from left-to-right; in Thai, characters may be written above, below, or to the right of the previous character; and, of course, in certain languages characters are written from top to bottom.

These space generation characteristics must be wholly independent of other processing functions. An application must function according to a "virtual" presentation model that consists of a succession of virtual lines. Each line is a succession of characters. The order of presentation of characters and lines are defined just before the materialization of these characters on the presentation device, within the presentation layer.

For each writing system, the following must be defined:

— The main script direction:

  — left-to-right

  — right-to-left

  — top-to-bottom

— Space generation must be defined for each character in relation to the main script direction:

  — positive space generation advances the current position in the main script direction.

  — negative space generation advances the set of characters behind the current position by one position in the main script direction but does not modify the location of the current position.

- zero (nil) space generation affects neither the current position nor the position of the characters behind the current position.

- neutral space generation corresponds to characters that present no specific space generation characteristics, but which follow the space generation direction established by previous characters.

- Certain characters may present different space generation characteristics depending on the context in which they are used: for example, the comma in Arabic may be considered as a text comma and be assigned a positive space generation, or be considered as a radix character in which case space generation is negative.

- During multi-lingual operations, there may be more than one main script direction, with several nested presentation modes.

To solve these problems, a general presentation model must ensure the complete independence of the logical order of those characters required for processing from the physical order necessary for the presentation of the characters.

When implementing specific space generation functions in an application using multi-lingual data, the aim must be to ensure that any required multi-lingual processing remains transparent to the user.

C.4 Justification

Justification methods may depend on the language: in Arabic for example, words are not divided and justification is achieved by adding connecting characters between two joinable characters.

C.5 Numeric representation

In certain languages, such as Arabic, where the direction of alphabetic characters is different from that of numeric characters, the representation of non-significant zeros by means of symbols other than spaces (for example a "digital" space) eliminates any ambiguity between an alpha-field space and a numeric field space sharing with the other characters in the field the space generation functions specific to numeric fields.

Annex D

Examples of solutions

D.1 The Canadian requirement for ordering English and French

One of the most important specification of cultural convention is the specification of the characteristics of ordering for text data strings. The first normative requirement for a comprehensive, fully predictable, culturally valid, requirement for ordering was the Canadian Standard CSA Z243.4.1-1992. This was adopted as a Canadian National Standard in 1994 after 6 years of work, which involved input from 7 different countries (Canada, France, U.S., Belgium, The Netherlands, Germany, Switzerland), and was based on a 1986 Québec government proposal.

The Canadian Standard describes collating weights which are usable for dictionary ordering of English, French, German, Dutch, Portuguese and Italian, and could be extended to other languages with only slight modification.

The technique described in the Standard assigns four levels of weights that can be used for fine-tuning the ordering function, and provides absolute predictability of results while being culturally acceptable to majority of the users of those languages. It is based mainly on ordering rules used in the main dictionaries of the French, English and German languages, and the primary rules learned at school by all young children, and unlike other more sophisticated classification techniques, can be understood by all, not only by scholars.

D.1.1 Example of cultural requirement

The rules are essentially as follows:

a) The three languages agree on a single alphabetic order, from A to Z, where no consideration is normally made for diacritical signs for the single purpose of ordering, unless there is a tie due to quasi-homography, i.e. words that look the same if diacritics are removed.

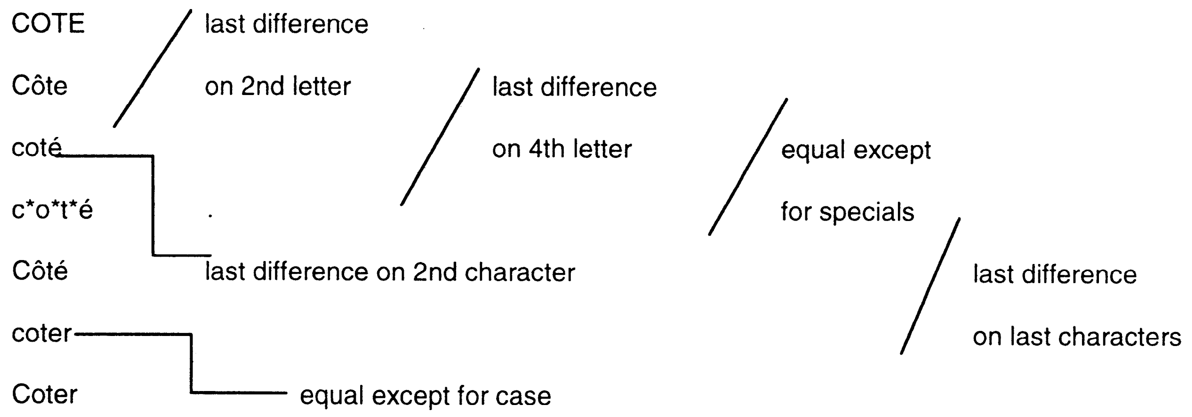
b) For some characters, expansion is carried out as if they were written as two separate letters (ae, oe, ss [and ij in Dutch for that matter]);

c) English and German dictionaries state that unaccented words precede accented ones in case of quasi-homography; French dictionaries need more precise rules as it quite common for lists of 3 or 4 quasi-homographs to be encountered for a series of identical letters, accented differently; in case of quasi-homography, the rule is in French is that "the last difference in the word determines the order" (which means scanning the words to be compared backwards from the end until a difference is encountered in accentuation). In order not to make French a special case, many sources recommended using the French rule for solving ties due to the fact that it is generally recognised that this does not create any extra overhead for other languages (since a stack is as easy to use as a list), and that this is culturally equivalent for other languages.

d) In case of homography on alphabetic characters and diacritics, then case becomes significant in determining a difference. English and German dictionaries agree that small letters should precede capital letters. French dictionaries do not make a difference, as they generally use only capital letters for their general language words (including accented capital letters, contrary to a wide-spread belief that capital accented letters are not used in French!). French encyclopaedias and proper name dictionaries tend to use capital letters first, but, as there are numerous exceptions and dictionaries are mute on this subject, it has been decided to use English and German rules in the Canadian Standard in order to harmonise rules without really making the rule culturally incorrect for French: it should, however, be noted that, for example, Danish dictionaries explicitly state that capital precede small letters.

e) Characters not part of the alphabet (spaces, hyphens, apostrophes, asterisks and so on, orthographic or not) are not significant for dictionary ordering. In order that ordering can be predictable, the Canadian standard specifies an order for these, but only for the case in which the other 3 levels of significance of text data (alphabetic data, diacritics and case) are absolutely identical. At present, this is the way the Canadian Standard specifies it, even though the normative benchmark contains a list of English words that could be ordered in a more refined way; however, no English-speaking native has objected and so it is assumed to be culturally acceptable. Thus "coop, co-op, COOP, CO-OP" constitutes a correctly sorted list according to the Canadian Standard; a refinement might have been to make "coop, COOP, co-op, CO-OP" the preferred order, but this seems to be a matter of personal preference and of very fine tuning that could cause an extra overhead in some environments (requiring 5 levels instead of 4). It might, however, be more consistent for certain very specific, orthographic characters like hyphens, apostrophes and spaces to be treated specially for all languages — which would involve these characters being processed after the diacritics have been considered but before case, the other special characters being processed after case.

For example, the following records are ordered correctly per the Canadian Standard specification:


COTE
Côte
coté
c*o*t*é
Côté
coter
Coter
last difference on 2nd letter
last difference on 4th letter
equal except for specials
last difference on last characters
last difference on 2nd character
equal except for case

D.1.2 An example of a specification technique

The Canadian Standard describes this behaviour using words (in English and in French), diagrams, and tables. To simplify the understanding of such tables, consider the following tables of relative numbers:

	ALPHA	ACCENTS	CASE	SPECIALS
	1st level	2nd level	3rd level	4th level
INPUT	token	token	token	token
CHARACTER	(serial)	(stacked)	(serial)	(serial)
c	6	3	1	N/A
C	6	3	2	N/A
e	7	3	1	N/A
é	7	4	1	N/A
E	7	3	2	N/A
o	8	3	1	N/A
ô	8	5	1	N/A
O	8	3	2	N/A
r	9	3	1	N/A
t	10	3	1	N/A
T	10	3	2	N/A
*	N/A	N/A	N/A	1

The Canadian Standard then suggests a conformance algorithm that establishes a series of subkeys to be numerically composed for our examples as follows (where the numbers are only indicative here, and show a relation only between the subset of characters chosen for the purpose of the example). To avoid composing a fourth key with place holders for all non-special characters, comparison is done on the positions of the special characters, and in case of equality, on the weight assigned to the special character itself. For that algorithm to work if all 4 subkeys are concatenated for a multi-level one-pass sort, a special logical zero delimiter (which could be 1 for C if all other relative numbers are offset by 1!) is coded between the 3rd and 4th subkey. If certain conditions are not met in the careful choice of relative numbers, such a logical zero delimiter would be advisable between each of the subkeys.

Original string	Subkey 1	Subkey 2	Subkey 3	Logical	Subkey 4
		delim.	align		
COTE	6,8,10,7	3,3,3,3	2,2,2,2	0	
Côte	6,8,10,7	3,3,5,3	2,1,1,1	0	
coté	6,8,10,	4,3,3,3	1,1,1,1	0	
c*o*t*é	6,8,10,7	4,3,3,3	1,1,1,1	0	2,1,4,1,6,1
Côté	6,8,10,7	4,3,5,3	2,1,1,1	0	
coter	6,8,10,7,9	3,3,3,3,3	1,1,1,1,1	0	
Coter	6,8,10,7,9	3,3,3,3,3	2,1,1,1,1	0	

The Canadian Standard presents a non-normative reduction technique (originating from the Québec government department which implemented it [designer: Alain laBonté]) to reduce these subkeys without affecting the comparison process if keys are to be stored by an application for further comparison by a dump process (such as a hardware device able to search on binary sequences or an old unmodifiable "indexed sequential" access method of any kind that orders keys numerically). The net effect is that for most French words, and more than 99% of English words, no storage is required for the second key, as no accent is present, and in certain conditions storage is highly reduced for the third subkey. The same subkeys reduced and concatenated to give a one-pass directly comparable numerically would be:

COTE	6,8,10,7		2,2,2,2	0	
Côte	6,8,10,7	3,3,5	2	0	
coté	6,8,10,7	4		0	
c*o*t*é	6,8,10,7	4		0	2,1,4,1,6,1
Côté	6,8,10,7	4,3,5	2	0	
coter	6,8,10,7,9			0	
Coter	6,8,10,7,9		2	0	

Note that this reduction technique should not be implemented without first carefully looking at the Canadian standard, and its references, for caveats in designing other tables. Only in certain conditions can such an optimising technique be used. However, even without reduction, the principle of forming a single key (out of a multi-level specification) to be passed to old applications that "know" how to sort numerical data strings is highly valid and economically very important for the support of existing applications that can be "internationalized" without significant modifications, if any.

D.2 Other specification techniques

After Canada released its specification, POSIX defined a model that could handle it in a general way. Although this is an abstract specification technique, it nevertheless does the job adequately. A good recommendation would be not to reinvent the wheel, but to use this technique, as it is the only international specification technique currently in existence for describing collation tables. It does not use relative numbers but, instead, uses a clever sequential ordering system that allows the description of multi-level weights without specifying any numerical data. Ordering can be changed simply by inserting lines for specific additional characters or by swapping lines.

Moreover it is, like the Canadian Standard, a coded character set-independent specification technique which will not necessitate as many specifications as there are equivalent character sets.

Hence it is a very flexible technique that allows the handling of a general specification. It may be that refinements will be made in the future, but, like the Canadian Standard, it represents the state of the art in this domain, and it is expected that future work will build on this specification technique. Current expert opinion is that it can handle most of the languages and scripts of the world without any major difficulties. Extensions are conceivable to handle combining sequences, as present in ISO/IEC 10646 implementation level 3, for those languages that absolutely require those combinations to be handled to give a culturally valid ordering.

The specification of the minimum table to describe the previous example, according to the POSIX ordering specifications, would be (simplified):

```
...

collating-symbol <SMALL>

collating-symbol <CAPITAL>     Definitions of symbols that are not

collating-symbol <NONE> known  as characters but which are needed

collating-symbol <ACUTE>       to describe relative weights

collating-symbol <CIRCUMFLEX>

...


order_start forward;backward;forward;forward,position

                        |This statement

                        |describes the

                        |scanning direction

                        |for each level

                        |(even allows

                        |position tokens

                        |if desired)
```

```
<SMALL>                                           |will result in SMALL=1
<CAPITAL>                                         |will result in CAPITAL=2
<NONE>                                            |will result in NONE=3
<ACUTE>                                           |will result in ACUTE=4
<CIRCUMFLEX>                                      |will result in CIRCUMFLEX=5
<c>    <c>;<NONE>; <SMALL>                  |... c=6 and reused for itself
<e>    <e>;<NONE>; <SMALL>                  |... e=7 and reused for itself
<o>    <o>;<NONE>; <SMALL>                  |... o=8 and reused for itself
<r>    <r>;<NONE>; <SMALL>                  |... r=9 and reused for itself
<t>    <t>;<NONE>; <SMALL>                  |... t=10 and reused for itself
<C>    <c>;<NONE>; <CAPITAL>                | from now on,
<O>    <o>;<NONE>; <CAPITAL>                | no new value that needs to
<T>    <t>;<NONE>; <CAPITAL>                | be resolved; numeric weights
<E>    <e>;<NONE>; <CAPITAL>                | are all already known
<é>    <e>;<ACUTE>;      <SMALL>
<ô>    <o>;<CIRCUMFLEX>;<SMALL>
<*>    IGNORE;IGNORE;IGNORE;<SMALL>  |SMALL=1, why not?
                                                  |IGNORE means no value assigned
                                                  |at each level that specifies it
```

D.3 User group requirements and functionality

Interestingly enough, GUIDE SHARE Europe (used be SHARE Europe), having examined the Canadian specifications, described a series of programming requirements in a white paper published in 1990 in Geneva, entitled "National Language Architecture". Contrarily to the POSIX standards (ISO/IEC 9945-1 and ISO/IEC 9945-2) which, surprisingly, do not define the functions that could be associated with the specifications of ordering, GUIDE SHARE Europe requires the support of a series of functions at the operating system level to exploit to its full potential the specification of ordering.

It should be noted that if an operating system does not provide those functions they could be implemented in a common set of library routines available to different programming environments, and that is exactly what the Québec government has done for its data centres and is about to implement for small machines (personal computers, mini-computers, and so on) without any modification to the compilers it uses. For economic reasons, surprising as it may seem, COBOL has been used for this purpose, in spite of the recommendation to use a more portable programming language. For environments other than the mainframes, other decisions have been taken (and C language routines are being developed).

Obviously if some of these functions were implemented in programming language syntax, development of applications would be easier for programmers, and there might also be some gains in performance.

D.3.1 GUIDE SHARE Europe Requirements of functions

The following requirements have been addressed by GUIDE SHARE Europe in the above mentioned "White Paper on National Language Architecture":

D.3.1.1 Extended key generation

Given a string and identification of its coding, a function should exist to return the 4 subkeys of the Canadian specification (note: this could be generalised to N levels instead of 4, with a possible information being returned on the number of levels and a table of dimension N for the N subkeys).

D.3.1.2 Original key regeneration

Given the N keys generated by the previous function, it should be possible to regenerate the original in the coding specified (coding which could be different from the original, although the original character string would be functionally equivalent to the original from the user's point of view). This supposes that the system of tables used to generate extended subkeys is known to the underlying process, of course. Since all the information is contained in the extended subkey (principle of absolute predictability), this has been shown to be possible and has been implemented by the Québec government purely for the requirements of the Canadian specification.

D.3.1.3 Comparison operation

Given 2 character strings on input and their coding, or the N subkeys, return the following information:

Case 1: The 2 strings are absolutely equal (ex. "ABC"="ABC");

Case 2: The 2 strings are equivalent up to the level N of comparison;

Case 2.1 Canadian spec (ex. "COTE"=="Coté at level 1 only);

Case 2.2 Canadian spec (ex. "cote"=="Cote" up to level 2);

Case 2.3 Canadian spec (ex. "c*o*t*e"=="cote" up to level 3)

Case 3: String 1 comes before string 2 in order (ex. "Cote"<"coté)

Case 4: String 1 comes after string 2 in order (ex. "coté" >"COTE")

Case 5: Fuzzy match: "Phydeault" ~ "FIDO" for French (snobbish dogs obviously write their names using the first spelling!)

NOTE D1 Case 2 is a rephrasing of the GUIDE SHARE Europe requirement: the original requirement specifies on input what kind of equivalence is accepted, the answer indicating equality only for this case if absolute equality is not returned, i.e. equivalence required if only different because of specials, or because of case, or because of diacritics. This is better generalised to N levels with this respecification.

NOTE D2 Case 5 requires algorithmic fuzzy pattern matching functions that go beyond economical development in most environments because they require expert system technology which, for example, "knows" the exact phonetic environment in which it is applied. Examples of this might be establishing the phonetic equivalent of cockney English (as spoken in certain parts of London, but not elsewhere), or of foreign accent biases on a language, etc. At the time at which the Canadian Standard was being developed this functionality was commercially available in pedagogic applications (for teaching French to young children of different origins) for applying both the Québecer accent and the various foreign accents commonly encountered in Montreal to French. However, the cost of implementing this requirement, which might be useful for a police department but not for most commercial applications, would have more than tripled the cost of providing the basic functions mentioned above, and it was decided not to incorporate it within the Standard.

D.3.1.4 Coding conversion

Given a string and its coding, and a resulting coding identification, return an equivalent string in its new coded equivalent.

D.3.1.5 Sort

Given a list of strings, perform an internal sort using the comparison operation described above to obtain consistent results. For an external sort, the same function should be used to obtain the same consistency of operation.

D.3.1.6 Merge

Given two lists assumed to be sorted according to the previous function, merge the two lists in one using the same comparison operation described above.

D.3.1.7 Substring Search

Given two strings, search for the occurrence of the second one in the first one, with parameters indicating what kind of equivalence level is acceptable; return the offset of the retrieved string and its length (which can be different from the one searched if equivalences are encountered, as for example if ligatures are equivalent to separate letters in a given specification).

D.3.1.8 Conversion to upper case unaccented data

Given a rich text data including accented/unaccented lower/upper-case data, return the "traditional" unaccented upper-case equivalent (see below under clause D.4 on how the at-first-glance-unrealistic reciprocal function has been implemented, even if it is not a requirement so far in the international community).

D.4 Complementary functions out of the scope of programming languages

The implementation of these functions from scratch will be very useful in all new end-user environments (including U.S. sites, some of which are said to have also implemented the Canadian specifications as it was considered a requirement to solve problems of character data processing in a monolingual English environment).

However it may be interesting to know that old data bases in Québec, as in Europe in general, have used unaccented-capital-letters-only data for a long time to avoid many of the problem solved by these specifications (although not all, as the presence of special characters, even if less visible, is an existing problem, with varying degrees of seriousness). To implement these new functions, mixing upper-lower-accented-unaccented data is necessary for communicating with external sites. The requirements of GUIDE SHARE Europe allow such mixing. Furthermore the Québec government also implemented automatic functions to add accents and lower-case to existing personal data and geographical name data that was unaccented before (with a 99,7% accuracy, the remaining cases necessitating human intervention because they were unsolvable by automatic means, such as homographs like "Masse" and "Massé, two existing family names), in order to avoid having to re-type that information in huge data bases. These are problems for which no requirement is likely to be addressed to programming language standards designers, but which are nevertheless real and solvable in existing environments, thus demonstrating that it is also possible to deal with the past without the necessity of starting from scratch (in which case no action would ever be possible!).

D.5 Consequences of imbedding these functions in languages

The basic functions that were previously mentioned are implementable using present tools or with extensions to existing languages. The latter is highly desirable, in order that the resulting programs can be designed to be portable in different cultures, the behaviour of the functions being parametrically provided outside of the language, but the functionality being fully provided by the language to directly interface those external specifications, while optimising performance goals.

D.6 A specific language implementation example

The following example indicates how a Fortran module might be used to implement culturally sensitive string comparison, using the approach outlined above. This module assumes the existence of the intrinsic functions suggested in 7.5, together with two additional intrinsic functions `GENERATE_KEYS` and `COMPARE_STRINGS` which provide the functions described in D.3.1.1 and D.3.1.3, respectively.

```
MODULE Cultural_Strings
   IMPLICIT NONE
   PRIVATE

   !  This module provides the necessary services to support the
   !  character handling requirements in a particular model of
   !  INTERNATIONALIZATION and LOCALIZATION.

   !  As written here it operates automatically in the cultural
   !  environment which is current when the program begins
   !  execution. It could be extended to allow for a user-specified
   !  cultural environment.

   !  Establish current cultural environment and character kind
   INTEGER, PARAMETER :: environment=CULTURAL_ENVIRONMENT(),  &
                         ch_kind=REPERTOIRE_KIND()

   !  Specify overloaded comparison operators
   INTERFACE OPERATOR ( < )
      LOGICAL FUNCTION cultural_lt (s1,s2)
         CHARACTER(KIND=ch_kind,LEN=(*)), INTENT(IN) :: s1,s2
      END FUNCTION cultural_lt
   END INTERFACE

   INTERFACE OPERATOR ( <= )
      LOGICAL FUNCTION cultural_le (s1,s2)
         CHARACTER(KIND=ch_kind,LEN=(*)), INTENT(IN) :: s1,s2
      END FUNCTION cultural_le
   END INTERFACE
   .
   .
   !  Specify those entities to be exported from the module
   PUBLIC environment,ch_kind,OPERATOR(<),OPERATOR(<=), ...
```

```
CONTAINS

  LOGICAL FUNCTION cultural_lt (s1,s2)
    CHARACTER(KIND=ch_kind,LEN=(*)), INTENT(IN) :: s1,s2
       INTEGER :: key1(LEN(s1),4),key2(LEN(s2),4)
    LOGICAL :: comp(5)


    ! The intrinsic function GENERATE_KEYS takes a character
    ! string, and returns the four subkeys of the Canadian
    ! Standard for each character as a rank two integer array of
    ! dimension four by the number of characters in the string.
    key1 = GENERATE_KEYS(s1)
    key2 = GENERATE_KEYS(s2)

    ! The intrinsic function COMPARE_STRINGS takes two arrays
    ! of integer subkeys and returns a rank one logical array
    ! of dimension 5. Each element of the array value of the
    ! function specifies the truth or otherwise of the
    ! corresponding case in the specification of the comparison
    ! operation.
    comp = COMPARE_STRINGS(key1,key2)

    ! Return result of comparison as true if Case 3 is true (s1
    ! before s2) and Cases 1 and 2 are false (s1 not equal and
    ! not equivalent to s2)
    cultural_lt = comp(3) .AND. .NOT.(comp(1) .OR. comp(2))

  END FUNCTION cultural_lt

  LOGICAL FUNCTION cultural_le (s1,s2)
    CHARACTER(KIND=ch_kind,LEN=(*)), INTENT(IN) :: s1,s2
    INTEGER :: key1(LEN(s1),4),key2(LEN(s2),4)
    LOGICAL :: comp(5)
```

```
      key1 = GENERATE_KEYS(s1)

      key2 = GENERATE_KEYS(s2)

      comp = COMPARE_STRINGS(key1,key2)


      !  Return result of comparison as true if any of Case 1

      !  (s1 equals s2), Case 2 (s1 equivalent to s2) or Case3

      !  (s1 before s2) is true

      cultural_lt = comp(1) .OR. comp(2) .OR. comp(3)


   END FUNCTION cultural_lt

      .

      .

END MODULE Cultural_Strings
```

A program which wished to use this module to provide culturally correct character handling could do so as follows:

```
PROGRAM Culturally_correct
   !  Obtain access to all public elements in Cultural_Strings
   USE Cultural_Strings
   IMPLICIT NONE
   !  Declare two 50-character strings of the default type for the
   !  current environment
   CHARACTER(KIND=ch_kind,LEN=50) :: string_1,string_2
   !  Read data into these strings
   READ *,string_1,string_2
   !  Print the two strings in their correct order, using the
   !  overloaded <= operator to ensure culturally correct ordering,
   !  with the first input coming first if they are equal, or at
   !  least equivalent
   IF (string_1 <= string_2) THEN
      PRINT *,string_1,string_2
   ELSE
      PRINT *,string_2,string_1
   END IF
     .
     .
END PROGRAM Culturally_correct
```

ICS 35.060

Descriptors: data processing, information interchange, network interconnection, programming languages, implementation, models.

Price based on 47 pages