



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.781

(08/2001)

SERIES X: DATA NETWORKS AND OPEN SYSTEM
COMMUNICATIONS

OSI management – Management functions and ODMA
functions

**Requirements and guidelines for
Implementation Conformance Statements
proformas associated with CORBA-based
systems**

ITU-T Recommendation X.781

ITU-T X-SERIES RECOMMENDATIONS
DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.369
IP-based networks	X.370–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	X.680–X.699
OSI MANAGEMENT	
Systems Management framework and architecture	X.700–X.709
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	X.730–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation X.781

Requirements and guidelines for Implementation Conformance Statements proformas associated with CORBA-based systems

Summary

This Recommendation specifies the Implementation Conformance Statements (ICS) proforma of CORBA-based system interfaces which will be used in telecommunication network management, and also provides the testing methodology for CORBA-based interfaces.

Source

ITU-T Recommendation X.781 was prepared by ITU-T Study Group 4 (2001-2004) and approved under the WTSA Resolution 1 procedure on 13 August 2001.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

© ITU 2002

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from ITU.

CONTENTS

	Page
1 Scope.....	1
2 References.....	1
3 Definitions	1
3.1 Implementation conformance statements definitions	1
4 Abbreviations and acronyms	2
5 Basis of conformance test methodology for CORBA-based management system interface	2
5.1 Overview.....	2
5.2 Methodology of conformance test for CORBA-base interfaces.....	2
6 CORBA-based management interface ICS proforma.....	3
6.1 IDL overview	3
6.2 Requirements and guidelines for specification of CIICS proformas.....	4
6.2.1 General instructions for CIICS proforma specification.....	4
6.2.2 Interface support proforma	5
6.2.3 Attribute support proforma.....	6
6.2.4 Operation support proforma	6
6.2.5 Data type support proforma.....	8
7 Instructions for completing the CIICS proforma.....	8
7.1 Definition of supported.....	8
7.2 The Status column	8
7.3 The Support column.....	9
7.4 The "Constraints and Values" column.....	9
7.5 The "Additional information" column.....	10
7.6 The "Index" column.....	11
7.7 The "Subindex" column.....	11
7.8 The "Category" column	11
7.9 The "Field property" column	11
7.10 The "Interface identifier", "Attribute identifier", "Operation identifier" columns	11
7.11 The "Derived interface" column.....	11
Annex A – Example of CIICS proforma specification.....	12
A.1 CORBA IDL definition	12
A.2 CIICS	12
Annex B – Example showing "Data type support proforma" specification.....	15
B.1 CORBA IDL definition	15

	Page
B.2 CIICS	16
Annex C –Example showing "Appendix any type support proforma" specification.....	18
C.1 CORBA IDL definition	18
C.2 CIICS	19
Annex D –Example showing "valuetype support proforma" specification	20
D.1 CORBA IDL definition	20
D.2 CIICS	21

ITU-T Recommendation X.781

Requirements and guidelines for Implementation Conformance Statements proformas associated with CORBA-based systems

1 Scope

This Recommendation provides the requirements and guidelines for CORBA Interface Implementation Conformance Statement (CIICS) proforma and the specification of this proforma. The CIICS is a statement made by an implementer to claim conformance to a CORBA/IDL based interface definition.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

- ITU-T X.296 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – Implementation conformance statements*.
- ITU-T X.724 (1996), *Information technology – Open Systems Interconnection – Structure of management information: Requirements and guidelines for implementation conformance statement proformas associated with OSI management*.
- ISO/IEC 9646-7:1995, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 7: Implementation Conformance Statements*.
- OMG (1998), *The Common Object Request Broker: Architecture and Specification, Revision 2.3*.

3 Definitions

3.1 Implementation conformance statements definitions

This Recommendation makes use of the following terms defined in ITU-T X.296 and ISO/IEC 9646-7:

- a) (ICS) item;
- b) (ICS) question;
- c) status (value);
- d) (support) answer.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations:

CIICS	CORBA-based Interface Implementation Conformance Statement
CORBA	Common Object Request Broker Architecture
ICS	Implementation Conformance Statement
IDL	Interface Definition Language
IUT	Implementation Under Test
IXIT	Implementation Extra Information for Testing
ODP	Open Distributed Processing

5 Basis of conformance test methodology for CORBA-based management system interface

5.1 Overview

Conformance relates an implementation to a standard. It states in which way systems, implemented with respect to a standard, can vary without errors occurring in their cooperation. If an implementation fulfils these requirements, then it is conform to the standard. The check of the statements is the conformance test. The starting point is the definition of conformance requirements in implementation independent interface specifications on the bases of an identification of reference points. A management interface specification should define conformance reference points at which an object must be tested to check if it fulfils a set of conformance criteria. During the test, a number of stimuli and events are observed and evaluated at these conformance points.

Management interface specifications should include conformance statements which identify conformance reference points at every interface of the specified objects.

Because in general the information flow between two system components is realized through several reference points, the conformance test has to take into consideration:

- a) the test of such information flow at each reference point; and
- b) the test of consistency between the combinations.

So, a coordinated test at all identified reference points is necessary.

5.2 Methodology of conformance test for CORBA-base interfaces

There are static and dynamic conformance test requirements.

- a) Static conformance test requirements state the functionality which is at least necessary for conformance testing. Basis of the static conformance test is a so-called ICS proforma in which functional limitations and possibilities of the standard are defined. It is a document in form of a questionnaire which has to be answered by the implementer.
- b) Dynamic conformance test requirements specify the whole potential behaviours of an implementation visible at the identified reference points. The dynamic conformance test contains the realization of test cases in a test system.

Test cases include concrete test cases and abstract test cases. The concrete test cases are derived from abstract test cases which have to be given/defined by the standard. A basic precondition for the derivation of the test cases is the definition of test purposes by the standard.

Before starting the test in a test laboratory, the following steps must be processed:

- a) The implementer has to fill out the given ICS proforma. Information which makes the ICS proforma statements more precise or concerns the test realization is taken up in IXIT (implementation extra information for testing).
- b) The so-called real (concrete) test cases should be derived from the abstract test cases of the standard, or they can be derived from the test purposes directly, where generic things (types, etc.) are realized depending on the individual implementation.

Figure 5-1 illustrates this approach.

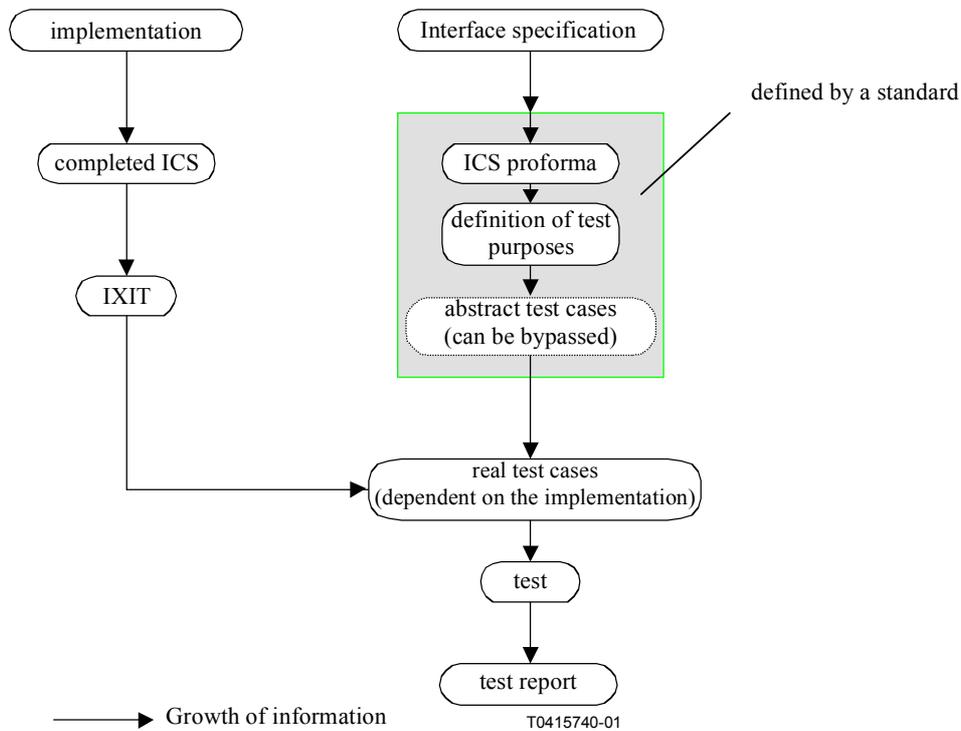


Figure 5-1/X.781 – Testing process of CORBA-based interfaces

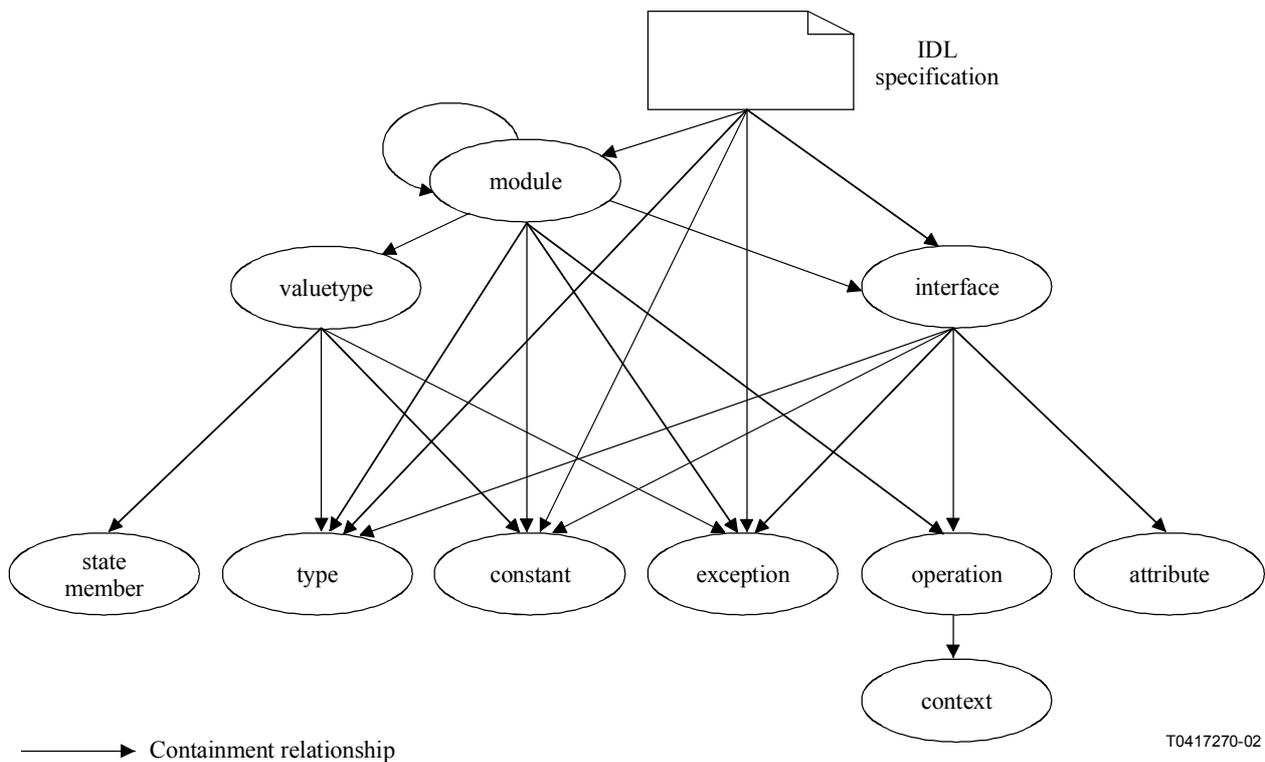
6 CORBA-based management interface ICS proforma

The CORBA Interface Definition Language (IDL) is used to define interfaces of objects in CORBA-based systems. The CIICS proforma must express the following IDL features such as IDL modules, interfaces and other IDL types. This clause introduces the features of CORBA IDL firstly, then specifies the CIICS proforma according to these language features.

6.1 IDL overview

The OMG Interface Definition Language (IDL) is the language used to describe the interfaces that client objects call and object implementations provide. An interface definition written in OMG IDL completely defines the interface and fully specifies each operation's parameters. An IUT may include an interface's client function or an interface's server function. So in the interface proforma there should be a space available to specify which side of function (client or server) the IUT implements.

The OMG IDL grammar is very similar to the proposed ANSI C++ standard. It has constructions such as module, interface, constant, type, operation, attribute, exception, valuetype, state member and context declarations. According to the IDL syntax specification, the containment relationship between these constructions can be illustrated as in Figure 6-1.



T0417270-02

Figure 6-1/X.781 – Containment relationship in IDL specification

The ICS proformas must represent the relationship shown above.

6.2 Requirements and guidelines for specification of CIICS proformas

Proforma specifications shall follow the style as documented in the following clauses. Proforma specifications shall provide the information required by this Recommendation. Additional tables may be included for other information, if needed.

There are three levels of documentation pertaining to CIICS, namely:

- a) guidelines or Recommendation tools for the production of CIICS proformas;
- b) a CIICS proforma, associated with a standard related to CORBA-based network management, which is to be filled in by a supplier of the implementation and when filled in is a CIICS;
- c) a CIICS prepared by a supplier of the implementation as part of a conformance claim to a standard related to CORBA-based network management.

6.2.1 General instructions for CIICS proforma specification

This Recommendation provides instructions to construct CORBA-based management system ICS proforma specification. CIICS proformas include four sub proformas: interface support proforma, attribute support proforma, operation support proforma and data type support proforma. All these proformas are in a tabular form which are similar to the proformas specified in ITU-T X.724.

Clause 6.2.2 describes the CIICS proforma for interfaces. Clause 6.2.3 describes the CIICS proforma for attributes. Clause 6.2.4 describes the CIICS proforma for operations. Clause 6.2.5 describes the CIICS proforma for complex data types. Annexes A, B, C and D provide examples of CIICS proforma specification, which is to be filled in by a supplier of an implementation.

The following common notations, defined in ITU-T X.291 and ISO/IEC 9646-2 and ITU-T X.296 and ISO/IEC 9646-7, are used for the "status" value column in this Recommendation:

- m Mandatory
- o Optional
- Not applicable or out of scope

NOTE – The notations "m", "o" are prefixed by a "c:" when nested under an optional item of the same table.

The following common notations, defined in ITU-T X.291 and ISO/IEC 9646-2 and ITU-T X.296 and ISO/IEC 9646-7 are used for the support answer column:

- Y Implemented
- N Not implemented
- No answer required
- Ig The item is ignored (i.e. processed syntactically but not semantically)

The CIICS proforma specification is formed by copying clauses 6.2.2, 6.2.3 6.2.4 and 6.2.5, completing the tables except for the "Support" and "Additional information" columns, and extending the remaining tables to meet the requirements of the specification. CIICS proforma shall provide tables for all the attributes, operations, parameters, exceptions and contexts defined in IDL information model, no matter derived from super-classes or added by redefinition.

To form a CIICS from a CIICS proforma, the supplier of the implementation shall fill in the "Support" and, if appropriate, "Additional information" columns of all the tables in the CIICS proforma.

6.2.2 Interface support proforma

The purpose of the proforma for interface is to provide a mechanism for a supplier of an implementation who claims conformance to an interface specification to provide conformance information in a standard form.

The interface support proforma is like Table 6-1:

Table 6-1/X.781 – Interface support table

Index	Interface identifier	Derived interface	Status	Support	Additional information

where:

- "Index" field is made up of a consecutive number for readers of the CIICS to refer to each item. Each interface defined in an IDL information model is given a unique number as its index.
- "Interface identifier" field is the absolute name of the interface, which consists of the name space and the interface name.
- "Derived interface" field is provided for the direct parent interface(s) the current interface derived from, if there are any.
- For each instantiable interface, if it is mandatory, the "Status" should be filled with 'm'; if it is optional, the "Status" should be filled with 'o'. If an interface is not instantiable, then the "Status" should be filled with "-", meaning not applicable.

- The "Additional Information" field is used to indicate in which side (client or server) function the IUT is implemented.¹

6.2.3 Attribute support proforma

The purpose of the proforma for attribute is to provide a mechanism for a supplier of an implementation who claims conformance to attributes specification in an interface to provide a standard form.

The attribute support proforma is like Table 6-2, which is to be implemented per IDL interface:

Table 6-2/X.781 – Attribute support table

Index	Attribute identifier	Constraints and values	Get		Set		Additional information
			Status	Support	Status	Support	

where :

- "Index" is the unique reference within an interface.
- The "Attribute identifier" is the relative name of the attribute.
- The "constraints and values" field is the data type description of the attribute.
- Both "Get" and "Set" fields contain "Status" and "Support". The "Status" of "Get" column should always be filled with 'm'. If the attribute is a normal attribute, the "Status" of "Set" should be filled with 'm'; while the attribute is a "readonly" one, the "Status" should be filled by '-', meaning not applicable.
- The "Additional Information" field is to provide a space for the implementer to add more specific information about this attribute, such as default values when the object is created, or the reference index number when the type of this attribute is "any". See 7.5 for details.

For any attributes of complex types, the detailed information about the type is expanded in a table under the "attribute support table", which is called "data type support table" and will be discussed in 6.2.5.

6.2.4 Operation support proforma

The purpose of the proforma for operation is to provide a mechanism for a supplier of an implementation who claims conformance to operations specification in an interface to provide conformance information in a standard form.

The operation support proforma is formed by Tables 6-3 and 6-4, which are to be implemented per IDL interface:

¹ In most cases the IUT acts in a server role, but when reporting a notification or something, it may act as a client and send information through invoking an operation provided by another server located in the tester side.

Table 6-3/X.781 – Operation support table (Part 1)

Index	Operation identifier	Status	Support	Additional information

Where:

The "Operation identifier" field is the relative name of the operation.

The "Additional Information" is to provide a space for the implementer to add more specific information about the operation.

For each operation, if it is mandatory, the "Status" should be filled with 'm'; if it uses System Exceptions, which would typically be specified not in IDL but in associated behaviour statements, an explanation should be filled in the "Additional Information" field to specify the type of System Exceptions it uses; when System Exception NO_IMPLEMENT can be used to indicate the operation is not supported, the "Status" should be filled with 'o'; if it has user exceptions for conditional packages, the "Status" should be filled with 'c'.

Other columns have similar meaning as described in 6.2.2

Table 6-4/X.781 – Operation support table (Part 2)

Index	Subindex	Operation field identifier	Constraints and values	Category	Field property	Status	Support	Additional information

Where:

- "Index" field is provided for the index of each operation specified in Table 6-1.
- "Subindex" field is to describe the reference of each related item of the operation, which is formed by connecting the index of operation and a unique number with a separator '.'.
- The "Operation field identifier" field is to describe any identifiers which are related to the operation, including parameter names, context identifiers and exception names.
- The "Constraints and values" field is the data type description of each parameter, return value and exception defined in this operation. And the detailed information will be expanded in "data type support proforma", as will be described in 6.2.5.
- The "Category" field is to distinguish different operation field types, which are classified into two types: "parameter" and "termination", which will be described later in 7.8.
- The "Field property" field is a further description of the "Category" field, which describes the mode or detailed types of the operation field, as will be seen in 7.9.
- The "Additional Information" field is to specify some specific information the implementor shall state. When the type of a parameter is "any", a reference index number should be filled in this field. See 7.5 e) for details.

6.2.5 Data type support proforma

Data type support proforma is the description for all the user-defined data types used in the IDL information model. It is an expansion of "Attribute support proforma" and "Operation support proforma". "Data type support table" is subdivided into three kind of tables, which are "Parameter support subtable ", "Return value support subtable" and "Exception support subtable" respectively, which have the same format shown in Table 6-5:

Table 6-5/X.781 – Data type support proforma

Index	Subindex	Field identifier	Constraints and values	Status	Support	Additional information

Where:

- Index is the reference to each operation field identifier occurred in Table 6-1.
- Subindex is the reference to the child field contained in the super data type.
- The "Field identifier" field is to describe each child field name of a constructed data type.
- The "Constraints and values" field is the data type description of each child field. If the type of the field is a complex one, there will be more comments to show its nested child fields just under the row it occupies, this process will be continued until the field types become basic types predefined in CORBA Specification.
- The "Additional information" field is to specify some specific information the implementor shall state. When the type of a field is "any", a reference index number should be filled in this field. When the field is a state member of a valuetype, the publicity and inheritance of it shall be described in this field. See 7.5 e) for details.

7 Instructions for completing the CIICS proforma

This clause gives the instructions for completing each column defined in 6.2 and its subclauses.

7.1 Definition of supported

A capability is said to be supported if the Implementation Under Test is able to realize the specified functionality.

7.2 The Status column

This column indicates the level of support required for conformance to a specific IDL specification. The values are the ones defined in 6.2.1.

Guidelines for completing this field is as follows:

- a) For "interface support table", if it is mandatory, the "Status" should be filled with 'm'; if it is optional, the "Status" should be filled with 'o'. If an interface is not instantiable, then the "Status" should be filled with "-", meaning not applicable.

- b) For "Attribute support table", 'm' should be filled in the "status column" for all the "Get", and also for "Set" if the attribute is a normal one; '-' should be filled in this field for "Set" if the attribute has the proceeding keyword "readonly".
- c) For "Operation support table", 'm' should be filled in this field if the operation is necessary for the management functionality; otherwise it could be filled with 'o'. If the operation is mandatory, for each "operation field" of this operation the "status" column should be filled with 'm', otherwise it should be field with 'c:m'.
- d) For "data type support table", only when a type is of the type "union", could the "status" column of its child fields be filled with 'o', as unions are mainly used to indicate options, especially when there is only one branch with "TRUE" as its discriminating value. For all the other types 'm' should be filled in this column. Of course, when an item containing other types is optional, all the items it contains should add "c:" as a prefix for this field, as mentioned in 6.2.1.

7.3 The Support column

This column shall be completed by the supplier or implementer to indicate the level of implementation of each item. The available selections for this field are listed in 6.2.1.

The following are the guidelines for completing this field:

- a) If an item is claimed as "supported", all the mandatory items it contains must also be supported. Otherwise, the "support" column can just be filled with 'N'.
- b) If the "state" column of an item is filled with '-', the only selection for the corresponding "support" column is '-!'
- c) In the CIICS Proforma tables, every item marked with 'm' should be supported by the IUT.

7.4 The "Constraints and Values" column

The "Constraints and values" columns of the tables (which are to be filled in the proforma specification if applicable) contain the constraints and values of the specific item (e.g. attribute, operation field). This information may include, **if applicable**:

- a) any constraints regarding support of the specific item;
- b) specific values for attribute or operation parameters which are supported;
- c) the allowed types according to the standard specification.

When the item denotes a type, the type name of this item must be filled in this column. This rule is suitable for "Attribute support table", "Operation Support Table" and "Data Type support table".

The type name used to fill the column can be described using the following generating expression:

<type name> ::= <basic type name> | <user-defined type name>

<basic type name> ::= **void | short | unsigned short | long | unsigned long | long long | unsigned long long | float | double | long double | char | boolean | octet | string | TypeCode | objref | any | wchar | wstring | fixed**

<user-defined type name> ::= string that denotes an absolute type name

For each user defined data types, the "type kind" is also suggested to be filled in this column, enclosed in a pair of bracket.

Type kind is the abstract description of the IDL types. The kind of each basic type has the same name as its type name. The type names of all the user-defined types are: "struct", "union", "sequence", "array", "enum", "interface", "valuetype", "exception", "bstring"².

7.5 The "Additional information" column

This column shall be completed by the supplier or implementer to explain some specific information about the implementation which is not included in other columns. This Recommendation gives some general rules which are to be followed by suppliers.

- a) For "Interface support table", server or client role should be explicitly specified in this field. When acting as a server, "As server" shall be filled in this field; otherwise "As client" shall be filled in this field. When the role of an interface is determined, all the operations and attributes contained in this interface will act in the same role.
- b) For "Attribute support table", if an attribute has default values when the object is created, it is suggested to add the value in this field. If an attribute is readonly, and its initial value is given when the object is created, "set by create" is to be filled in this field. If an attribute is inherited from another interface, "Inherited" shall be filled in this field.
- c) For "Operation support table", if an operation is inherited from another interface, "Inherited" shall be filled in this field. If some parameter has default values, it shall be stated in the corresponding row of this field for the parameter. For each user-defined exception, the condition of throwing the exception shall be stated in this field.
- d) For "Attribute support table" and "Operation support table", if an attribute or operation is overloaded in a child interface, "overload" should be pointed out in this field.
- e) For "Attribute support table", "Operation support table" and "Data type support table", when the type of an field is "any", there should be a reference filled in this field, indicating the index number of an appendix table, in which the possible run-time types to be inserted into this "any" are fully described. The recommended appendix index number is the same as the index in the first column of the same row, except that a prefix character "A" is added. The "any type appendix support table" has the same format as the "data type support table". (See examples shown in Annex C for details.)
- f) For "Data type support table", when the field is a state member of a valuetype, the publicity and inheritance of this state member shall be described. The publicity may be "public state" or "private state"; and if this state member is inherited from its base type, an "inherited from <base valuetype name>" sentence shall exist in this field. Annex D shows an example of CIICS proforma for valuetypes.
- g) For each operation, if it uses System Exceptions, the type of the System Exceptions it uses should be explained in this field for the operation. And if the System Exception NO_IMPLEMENT can be used to specify "not supported", the corresponding "Status" should be filled with 'o'.
- h) For every CIICS table, If the supplier has something special to claim, this field could be used. And if the space is not enough, expanded tables could be added as part of the CIICS proforma.

² "bstring" means "bounded string". In IDL, "bounded string" specifies a string with length limitation. In CIICS, bstring <n> is used to describe this type, where "n" string is the length of the string.

7.6 The "Index" column

Each line within the CIICS proforma is numbered at the left-hand edge of the line. This numbering is included as a means of uniquely identifying all possible implementation details within the CIICS proforma.

The means of referring to individual responses is done by the following sequence.

- a) a reference to the super-clause of the item;
- b) the separating character "."
- c) a unique number.

An example of the use of this notation is illustrated in Annex A.

The "index" column exhibits the containment relationship between IDL syntax structures. The containment relationship is illustrated in 6.1 "IDL overview".

7.7 The "Subindex" column

This column has the same meaning and format as the "Index" column. It is also made up of consecutive numbers.

- a) For "Operation support table", the super-clause of "Subindex" is the prepositive column "Index" in the same row.
- b) For "Data type support table", the super-clause of "Subindex" is the index of its direct containing item, which could be either in the "Index" column or the "Subindex" column.

7.8 The "Category" column

This column in the "Operation support proforma" can be filled in using two kinds of choices, which are "parameter" and "termination". The "parameter" indicates that the "Operation field" is an operation's invocation parameter (including context), while the "termination" indicates that the "Operation field" is a type of termination situations.

7.9 The "Field property" column

This column corresponds to the "Category" column. If the "Category" field is "parameter", this column could be "in", "out", "inout" or "context", indicating the mode of the parameter. If the "Category" field is "termination", this column could be "success reply" or "exception". Where, "success reply" is the desired reply of the operation, which can be either a return value or "void"; "exception" indicates the operation is abort for some particular reason.

7.10 The "Interface identifier", "Attribute identifier", "Operation identifier" columns

These columns are to be filled in the names of corresponding interfaces, attributes, or operations.

NOTE – "Interface identifier" is the absolute name of the interface, which can be uniquely identified within the name space of the whole IDL information model. "Attribute identifier", "Operation identifier" are just relative names of the item, which has the name space just in its containing interface.

7.11 The "Derived interface" column

IDL Specification supports the inheritance relationship. This column leaves the space to the derived interfaces, if any.

NOTE – Only the direct parent interfaces are to be filled in this column, the ancestor interfaces are not required to be filled in this field.

ANNEX A

Example of CIICS proforma specification

A.1 CORBA IDL definition

In this annex, the CIICS proforma for "Interface support", "Attribute support" and "Operation support" will be illustrated. The following is a simple example of IDL definition:

```
// DESCRIPTION:
//
// source IDL file for simple BANK example
//
//*****
module BankModule {
//
// a simple description of a bank account
//
interface account {
    readonly attribute float balance;

    void makeLodgement (in float f);
    void makeWithdrawal (in float f);
};
//
// a simple description of a bank current account
//
interface currentAccount : account {
    readonly attribute float overdraftLimit;
};
//
// a bank simply manufactures accounts
//
// bank::reject is raised if a duplicate account name is seen
//
interface bank {
    exception reject {};
    account newAccount (in string name) raises (reject);
    currentAccount newCurrentAccount (in string name, in float limit)
        raises (reject);
    void deleteAccount (in account a);
};
};
```

A.2 CIICS

According to the above IDL definition, the CIICS for this simple IDL information model can be illustrated as in Tables A.1 to A.3:

Table A.1/X.781 – Interface BankModule::bank support

Index	Interface identifier	Derived interface	Status	Support	Additional information
1	BankModule::bank		m	Y	As server

Table A.1.1/X.781 – BankModule::bank attribute support

There is no attribute definition in bank interface.

Table A.1.2/X.781 – BankModule::bank operation support (as server)

Index	Operation identifier	Status	Support	Additional information
1.2.1	newAccount	m	Y	
1.2.2	newCurrentAccount	m	Y	
1.2.3	deleteAccount	m	Y	

Index	Subindex	Operation field identifier	Constraints and values	Category	Field property	Status	Support	Additional information
1.2.1	1.2.1.1	name	String	parameter	in	m	Y	
	1.2.1.2		BankModule::account (interface)	termination	success reply	m	Y	
	1.2.1.3	reason		termination	exception	m	Y	when the account object with the same name has existed
1.2.2	1.2.2.1	name	String	parameter	in	m	Y	
	1.2.2.2	limit	Float	parameter	in	m	Y	
	1.2.2.3		BankModule::currentAccount (interface)	termination	success reply	m	Y	
	1.2.2.4	reason		termination	exception	m	Y	when the current-Account object with the same name has existed
1.2.3	1.2.3.1	a	BankModule::account (interface)	parameter	in	m	Y	
	1.2.3.2		Void	termination	success reply	m	Y	

NOTE – If the invocation parameter has default value, it is illustrated in the "Constraints and values" column.

Table A.2/X.781 – Interface BankModule::account support

Index	Interface identifier	Derived interface	Status	Support	Additional information
2	BankModule::account		m	Y	As server

Table A.2.1/X.781 – BankModule::account attribute support

Index	Attribute identifier	Constraints and values	Get		Set		Additional information
			Status	Support	Status	Support	
2.1.1	balance	float	m	Y	-	-	default value is 0 when the object is created.

NOTE – Because the "balance" attribute is a "readonly" attribute, only the "Get Status" is mandatory, the "Set Status" is not applicable.

Table A.2.2/X.781 – BankModule::account operation support (as server)

Index	Operation identifier	Status	Support	Additional information
2.2.1	makeLodgement	m	Y	
2.2.2	makeWithdrawal	m	Y	

Index	Subindex	Operation field identifier	Constraints and values	Category	Field property	Status	Support	Additional information
2.2.1	2.2.1.1	f	float	parameter	in	m	Y	
	2.2.1.2		void	termination	success reply	m	Y	
2.2.2	2.2.2.1	f	float	parameter	in	m	Y	
	2.2.2.2		void	terminator	success reply	m	Y	

Table A.3/X.781 – Interface BankModule::currentAccount support

Index	Interface identifier	Derived interface	Status	Support	Additional information
3	BankModule::currentAccount	BankModule::account	m	Y	As server

Table A.3.1/X.781 – BankModule::currentAccount attribute support

Index	Attribute identifier	Constraints and values	Get		Set		Additional information
			Status	Support	Status	Support	
3.1.1	balance	float	m	Y	-	-	Inherited
3.1.2	overdraft Limit	float	m	Y	-	-	the value is set by create

NOTE – Because the "balance" attribute is Inherited from BankModule::account interface, "Inherited" is specified in the "Additional information"

If an attribute or operation is overloaded in a child interface, "overload" should be pointed out in the "Additional information" field.

Table A.3.2/X.781 – BankModule::currentAccount operation support (as server)

Index	Operation identifier	Status	Support	Additional information
3.2.1	makeLogement	m	Y	Inherited
3.2.2	makeWithdrawal	m	Y	Inherited

Index	Subindex	Operation field identifier	Constraints and values	Category	Field property	Status	Support	Additional information
3.2.1	3.2.1.1	f	float	parameter	in	m	Y	
	3.2.1.2		void	termination	success reply	m	Y	
3.2.2	3.2.2.1	f	float	parameter	in	m	Y	
	3.2.2.2		void	termination	success reply	m	Y	

The above is a very simple example which shows the CIICS proforma specification for "Interface support proforma", "Attribute support proforma" and "Operation support proforma". In Annex B, an example will be shown to illustrate the "Data type support proforma".

ANNEX B

Example showing "Data type support proforma" specification

B.1 CORBA IDL definition

This annex illustrates an IDL example containing several IDL user-defined complex data types, which could be used to show the format of "Data type support proforma". Here is the IDL definition:

```
module typeExample {
    typedef short shortArray[2][3];
    typedef sequence<short,4> shortSequence;
```

```

enum Color { red, blue, green};
exception ExceptionType {
    short    number;
    string   reason;
};

typedef short short_alias;

struct DataStructure {
    short_alias  s;
    float        f;
};

union UN_DS {
    short s;
    DataStructure ds;
};

interface typeInterface {
    UN_DS op (
        in      shortArray      p1_sA,
        in      shortSequence    p2_sS,
        inout   Color            p3_color,
        out     DataStructure     p4_UT
    )
        raises (ExceptionType)
        context ( "key1", "key2" ) ;
};
};

```

B.2 CIICS

According to the above IDL definition, the CIICS for this example information model can be illustrated in the following tables:

Table B.1/X.781 – typeExample::typeInterface interface support

Index	Interface identifier	Derived interface	Status	Support	Additional information
1	TypeExample::typeInterface		m	Y	As server

Table B.1.1/X.781 – typeExample::typeInterface attribute support

The interface typeExample::typeInterface has no defined attribute.

Table B.1.2/X.781 – typeExample::typeInterface operation support

Index	Operation identifier	Status	Support	Additional information
1.2.1	Op	m	Y	

Index	Subindex	Operation field identifier	Constraints and values	Category	Field property	Status	Support	Additional information
1.2.1	1.2.1.1	p1_sA	::typeExample::shortArray (array[2][3])	parameter	in	m	Y	
	1.2.1.2	p2_sS	::typeExample::shortSequence (sequence<4>)	parameter	in	m	Y	
	1.2.1.3	p3_color	::typeExample::Color (enum)	parameter	inout	m	Y	
	1.2.1.4	p4_UT	::typeExample::DataStructure (struct)	parameter	out	m	Y	
	1.2.1.5		key1	parameter	context	m	Y	
	1.2.1.6		key2	parameter	context	m	Y	
	1.2.1.7		::typeExample::UN_DS (union)	termination	success reply	m	Y	
	1.2.1.8	Exception Type	::typeExample::ExceptionType (exception)	termination	exception	m	Y

Parameter support subtable

Index	Subindex	Field identifier	Constraints and values	Status	Support	Additional information
1.2.1.1	1.2.1.1.1		Short	m	Y	
1.2.1.2	1.2.1.2.1		Short	m	Y	
1.2.1.4	1.2.1.4.1	s	::typeExample::short_alias (short)	m	Y	
	1.2.1.4.2	f	Float	m	Y	

NOTE – The child field of a union type must contain at least one item as its mandatory child field. Here we suppose the "s" field is mandatory, and "ds" is optional. In fact, whether a child field of type "union" is mandatory or optional depends on the semantics of the implementation.

Return value support subtable

Index	Subindex	Field identifier	Constraints and values	Status	Support	Additional information
1.2.1.7	1.2.1.7.1	S	Short	m	Y	
	1.2.1.7.2	Ds	::typeExample::DataStructure (struct)	o	Y	
	1.2.1.7.2.1	S	::typeExample::short_alias (short)	c:m	Y	
	1.2.1.7.2.2	F	Float	c:m	Y	

NOTE – The child field of a union type must contain at least one item as its mandatory child field. In fact, whether a child field of type "union" is mandatory or optional depends on the semantics of the implementation. Here the "f" field is supposed to be mandatory, and "ds" field is optional. So the "Status" of the items contained in "ds" is all prefixed by "c:", meaning that only when "ds" is supported, could the items "s" and "f" child field be mandatory.

Exception support subtable

Index	Subindex	Field identifier	Constraints and values	Status	Support	Additional information
1.2.1.8	1.2.1.8.1	number	short	m	Y	
	1.2.1.8.2	reason	string	m	Y	

ANNEX C

Example showing "Appendix any type support proforma" specification

C.1 CORBA IDL definition

This annex illustrates an IDL example containing an interface which has an attribute of type "any" and an operation with an any-typed parameter, which could be used to show the format of "Appendix any type support proforma". Here is the IDL definition:

```

Module anyExample {

    struct DataStructure {
        short    s;
        string   str;
    };

    interface anyInterface {
        attribute any anyAttr;
        void op ( inout any p1 ) ;
    };
};

```

C.2 CIICS

According to the above IDL definition, the CIICS for this example information model can be illustrated in the following tables:

Table C.1/X.781 – anyExample::anyInterface interface support

Index	Interface identifier	Derived interface	Status	Support	Additional information
1	anyExample::anyInterface		m	Y	As server

Table C.1.1/X.781 – anyExample ::anyInterface attribute support

Index	Attribute identifier	Constraints and values	Get		Set		Additional information
			Status	Support	Status	Support	
1.1.1	anyAttr	any	m	Y	m	Y	See appendix Table C.1.1.1

NOTE – Because the type of the "anyAttr" attribute is "any", the "Additional Information" field is filled with the appendix reference index number of the corresponding appendix table: "C.1.1.1", which is composed of an "A" followed by the index of the this attribute "1.1.1". Here suppose the possible run-time type allowed for this attribute is "unsigned short", "long" and "float".

Table C.1.2/X.781 – anyExample::anyInterface operation support

Index	Operation identifier	Status	Support	Additional information
1.2.1	op	m	Y	As Server

Index	Subindex	Operation field identifier	Constraints and values	Category	Field property	Status	Support	Additional information
1.2.1	1.2.1.1	p1	any	parameter	inout	m	Y	See appendix Table C.1.2.1.1
	1.2.1.2		void	termination	Success reply	m	Y	

NOTE – Because the type of the "p1" parameter is "any", the "Additional Information" field for this item is filled with the appendix reference index number of the corresponding appendix table: "C.1.2.1.1", which is composed of an "A" followed by the subindex of the this parameter "1.2.1.1". Here suppose the possible run-time type allowed for this parameter is "string", and "anyExample::DataStructure".

The following are the appendix tables for the above described any types.

Table C.1.2.1/X.781 – <::anyExample::anyInterface::anyAttr→ anyAttr any types support>

Index	Subindex	Field identifier	Constraints and values	Status	Support	Additional information
1	1.1		long	m	Y	
2	2.1		ushort	m	Y	
3	3.1		float	m	Y	

Table C.1.2.1.1/X.781 – <::anyExample::anyInterface::op → p1 any types support>

Index	Subindex	Field identifier	Constraints and values	Status	Support	Additional information
1	1.1		string	m	Y	
2	2.1		anyExample::dataStructure (struct)	m	Y	
	2.1.1	s	short	m	Y	
	2.1.2	str	string	m	Y	

NOTE – The "Appendix any type support table" has the same format as the "Data type support table", and the contents in this table are all the allowed types that can be inserted into this field.

ANNEX D

Example showing "valuetype support proforma" specification

D.1 CORBA IDL definition

This annex illustrates an IDL example containing valuetype definition with inheritance, and is defined as a parameter of an operation of another interface, which could be used to describe the "valuetype support proforma". Here is the IDL definition:

```

Module valueModule {
    valuetype BaseValue {
        public short id;
    };
    valuetype ValueExample : BaseValue {
        private string name;
        public long value;

        factory init(in string name, in long value);
        //method
        void local_op ();
    };

    interface aInterface {
        void remote_op (in ValueExample p1);
    };
};

```

D.2 CIICS

According to the above IDL definition, the CIICS for this example information model can be illustrated in the following tables:

Table D.1/X.781 – valueModule::aInterface interface support

Index	Interface identifier	Derived interface	Status	Support	Additional information
1	valueModule::aInterface		m	Y	As server

Table D.1.1/X.781 – valueModule::aInterface attribute support

The interface valueModule::ValueInterface has no defined attribute.

Table D.1.2/X.781 – valueModule::aInterface operation support

Index	Operation identifier	Status	Support	Additional information
1.2.1	remote_op	m	Y	As Server

Index	Subindex	Operation field identifier	Constraints and values	Category	Field property	Status	Support	Additional information
1.2.1	1.2.1.1	p1	ValueModule::ValueExample (valuetype)	parameter	in	m	Y	
	1.2.1.2		void	termination	Success reply	m	Y	

Parameter support subtable

Index	Subindex	Field identifier	Constraints and values	Status	Support	Additional information
1.2.1.1	1.2.1.1.1	Id	short	m	Y	Public state, inherited from valueModule::BaseValue
1.2.1.2	1.2.1.2.1	Name	string	m	Y	Private state
	1.2.1.2.2	Value	long	m	Y	Public state

NOTE – From the above tables, we can see that when transferred as parameters, valuetype is treated almost the same as struct, except that valuetype may contain states inherited from its base valuetypes. Since "id" is a public state member which valueModule::ValueExample inherited from valueModule::BaseValue, the "public state, inherited from valueModule::BaseValue" is filled in the "Additional information" field. And since "name" is an ordinary private state member of valueModule::ValueExample, the "private state" is filled in the "Additional information" field.

Although there are initializer operations "init()" and ordinary operations local_op() defined in this valuetype valueModule::ValueExample, they are not appearing in the ICS proforma, since these kind of operations are all local operations, which is out of the scope of interface implementation conformance statements.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems