Committee Draft ISO/IEC CD	
Date: **2004-09-16**	Reference number: ISO/JTC 1/SC 32**N1172**
Supersedes document SC 32Nxxx	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/IEC JTC 1/SC 32 Data Management and Interchange Secretariat: USA (ANSI)	Circulated to P- and O-members, and to technical committees and organizations in liaison for voting (P-members only) by: **2004-12-17** Please return all votes and comments in electronic form directly to the SC 32 Secretariat by the due date indicated.

ISO/IEC CD 20944-41:200x(E)

Title: Information technology — Metadata Interoperability & Bindings (MDIB) Part 41: C API binding

Project: 1.32.17.01.41.00

Introductory note: The attached document is hereby submitted for a three-month letter ballot to the National Bodies of ISO/IEC JTC 1/SC 32. The ballot starts 2004-09-16.

Medium: E

No. of pages: 16

Address Reply to: Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32, Pacific Northwest National Laboratory, 13667 Legacy Circle Apt H, Herndon, VA, 20171, United States of America

Telephone: +1 202-566-2126; Facsimile; +1 202-566-1639; E-mail: MannD@battelle.org

Reference number of working document: **ISO/IEC JTC1 SC32 N1172**

Date: 2004-09-10

Reference number of document: **ISO/IEC CD1 20944-41**
[Release Sequence #7]

Committee identification: ISO/IEC JTC1 SC32 WG2

SC32 Secretariat: US

Information technology — Metadata Interoperability and Bindings (MDIB) — Part 41: C API binding

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International standard
Document subtype: if applicable
Document stage: (30) Committee
Document language: E

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 20944-41 was prepared by Technical Committee ISO/IEC JTC1, *Information Technology*, Subcommittee SC32, *Data Management and Interchange*.

ISO/IEC 20944 consists of the following parts, under the general title *Information technology — Metadata Interoperability and Bindings (MDIB)*:

— *Part 01: Framework*

— *Part 02: Common vocabulary*

— *Part 03: Common provisions for conformance*

— *Part 04: Generic usage*

— *Part 05: Common data structures and services*

— *Part 06: Semi-structured aggregation*

— *Part 20: Common provisions for coding bindings*

— *Part 21: XML coding binding*

— *Part 22: DIVP coding binding*

— *Part 23: ASN.1 coding binding*

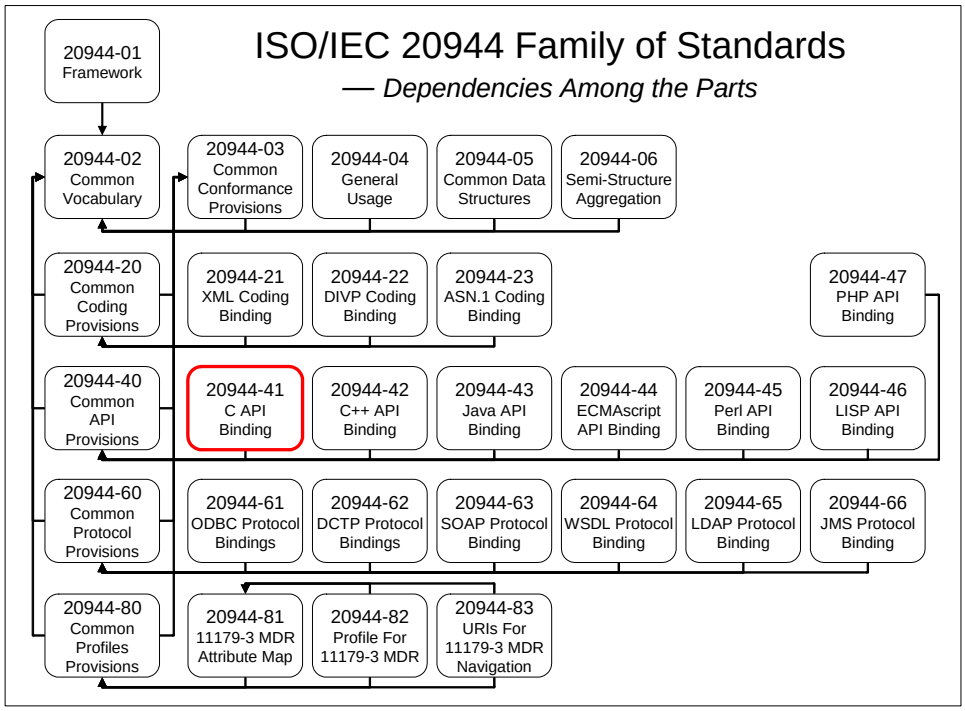
— *Part 40: Common provisions for application programming interface (API) bindings*

— *Part 41: C API binding*

- Part 42: C++ API binding
- Part 43: Java API binding
- Part 44: ECMAscript API binding
- Part 45: Perl binding
- Part 46: LISP binding
- Part 47: PHP binding
- Part 60: Common provisions for protocol bindings
- Part 61: ODBC protocol binding
- Part 62: DCTP protocol binding
- Part 63: SOAP protocol binding
- Part 64: WSDL protocol binding
- Part 65: LDAP protocol binding
- Part 66: JMS protocol binding
- Part 80: Common provisions for profiles
- Part 81: Attribute mapping for 11179-3 metadata registry metamodel
- Part 82: Profile for 11179-3 metadata registry metamodel
- Part 83: Uniform Resource Identifier (URI) suffixes for 11179-3 metadata registry metamodel navigation

Introduction

The following diagram shows the organization of the ISO/IEC 20944 family of standards with this Part highlighted.



Organization of ISO/IEC 20944 family of standards.

This Part of ISO/IEC 20944 specifies the C API binding, based upon the common API binding specified in Part 40. For example, common features include:

Information technology — Metadata Interoperability and Bindings (MDIB) — Part 41: C API binding

Editor's Note: Each part of 20944 is marked with a common sequence number ("**[Release Sequence #N]**") to indicate they are synchronized and harmonized among themselves. The mark "**[Release Sequence #N]**" does *not* imply that there are a complete set of N-1 prior drafts for any particular Part.

1 Scope

This Part of this International Standard specifies the C programming language API binding for the 20944 family of standards.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 20944-02:—[1], *Information technology — Metadata Interoperability and Bindings (MDIB) — Common vocabulary*[2]

ISO/IEC 20944-40:—[3], *Information technology — Metadata Interoperability and Bindings (MDIB) — Common provisions for application programming interface (API) bindings*

[1] To be published.

[2] The international standards, technical reports, and drafts of the 11179, 20943, and 20944 series are available at

http://metadata-standards.org/11179
http://metadata-standards.org/20943
http://metadata-standards.org/20944

[3] To be published.

3 Terms and definitions

The 20944 family of standards consolidates its terminology into a single part. The terminology of Part 02 is included via normative reference.

3.1
referenced data interchange specification
data model that is being used for a defined interoperability binding

NOTE The term *referenced data interchange specification*, defined in 20944-02, is used throughout the 20944 family of standards to reference the data model that is being used for the bindings. The *referenced data interchange specification* is tied to the bindings via normative reference, e.g., some other standard defines a data model and uses 20944, via normative reference, to provide some coding, API, or protocol bindings. For Part 82, the *referenced data interchange specification* refers to the 11179-3 metamodel. Part 04 of this International Standard, explains how other standards and specifications may use or re-use portions of the 20944 family of standards.

4 API mapping

The binding-independent API of Part 40 is mapped to the C programming language according to the following rules.

4.1 Datatype mapping

The datatypes of Part 40 are mapped as follows:

Part 40: Binding-Independent	Part 41: C API Binding
mdib_handle	mdib_handle
characterstring	string (maximum length 32767)
pointer	void *
str8	unsigned char []
str16	wchar_t[]
str32	wchar_t[]
int8, uint8, int16, uint16, int32, uint32, int64, uin64, int128, uint128	int8_t, uint8_t, int16_t, uint16_t, int32_t, uint32_t, int64_t, uint64_t, int128_t, uint128_t
real32, real64, real80	float, double, long double
procedure	(void *)()

4.2 Function parameter mapping

The parameters of Part 40 are the same as the parameters for this Part.

4.3 Function return mapping

The return values of Part 40 are mapped as follows:

Part 40: Binding-Independent	Part 41: C API Binding
state(success,failure)	int, success=0, failure=-1
characterstring	string (maximum length 65535)
str8	unsigned char []
str16	wchar_t[]
str32	wchar_t[]
int8, uint8, int16, uint16, int32, uint32, int64, uin64, int128, uint128	int8_t, uint8_t, int16_t, uint16_t, int32_t, uint32_t, int64_t, uint64_t, int128_t, uint128_t
real32, real64, real80	float, double, long double

4.4 Function exception mapping

The function exceptions of Part 40 are mapped as follows:

Part 40: Binding-Independent	Part 41: C API Binding
bad_conversion	errno is set

5 Procedure call signatures

5.1 Session establishment services

5.1.1 Connect

Synopsis

```
mdib_handle mdib_connect
(
    unsigned char *target, // repository to connect to
    unsigned char *options // connect options
)
```

5.1.2 Disconnect

Synopsis

```
int mdib_disconnect
(
    mdib_handle session // session handle
)
```

5.1.3 Open

Synopsis

```
mdib_handle mdib_open
(
    mdib_handle session, // session handle
    unsigned char *node, // portion of repository to open
    unsigned char *options // connect options
)
```

5.1.4 Close

Synopsis

```
int mdib_close
(
    mdib_handle session // session handle
)
```

5.2 Session parameter services

5.2.1 Get path

Synopsis

```
unsigned char *mdib_get_path
(
    mdib_handle session // session handle
)
```

5.3 Security services

5.3.1 Request Authorization/Authentication

Synopsis

```
int mdib_request_auth
(
    mdib_handle session, // session handle
    unsigned char *auth_type, // auth type
    unsigned char *auth_options // auth options
)
```

5.3.2 Response Authorization/Authentication

Synopsis

```
int mdib_response_auth
(
    mdib_handle session, // session handle
    unsigned char *auth_type, // auth type
    (*int)() auth_handler() // auth handler function
)
```

5.4 Data transfer services

5.4.1 Get value

Synopsis

```
int mdib_get_value
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier, // [in] src object name
    unsigned char *dst_label_type, // [in] saved label: typeof
    mdib_object *dst_label_ptr, // [out] saved label: ptr to
    unsigned char *dst_type_type, // [in] saved type: typeof
    mdib_object *dst_type_ptr, // [out] saved type: ptr to
    unsigned char *dst_object_type, // [in] saved value: typeof
    mdib_object *dst_object_ptr, // [out] saved value: ptr to
    unsigned char *dst_proplist_type, // [in] saved proplist: typeof
    mdib_object *dst_proplist_ptr, // [out] saved proplist: ptr to
)
```

5.4.2 Typed get value

Synopsis

```
unsigned char *mdib_get_value_as_str8
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

wchar_t *mdib_get_value_as_str16
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

wchar_t *mdib_get_value_as_str32
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

int8_t mdib_get_value_as_int8
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)
```

```
uint8_t mdib_get_value_as_uint8
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

int16_t mdib_get_value_as_int16
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

uint16_t mdib_get_value_as_uint16
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

int32_t mdib_get_value_as_int32
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

uint32_t mdib_get_value_as_uint32
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

int64_t mdib_get_value_as_int64
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

uint64_t mdib_get_value_as_uint64
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

int128_t mdib_get_value_as_int128
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

uint128_t mdib_get_value_as_uint128
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)
```

```
float mdib_get_value_as_real32
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

double mdib_get_value_as_real64
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)

long double mdib_get_value_as_real80
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name
)
```

5.4.3 Put value

Synopsis

```
int mdib_put_value
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier, // [in] src object name
    unsigned char *src_label_type, // [in] assigned label: typeof
    mdib_object src_label_ptr, // [in] assigned label: ptr to
    unsigned char *src_type_type, // [in] assigned type: typeof
    mdib_object src_type_ptr, // [in] assigned type: ptr to
    unsigned char *src_object_type, // [in] assigned value: typeof
    mdib_object src_object_ptr, // [in] assigned value: ptr to
    unsigned char *src_proplist_type, // [in] assigned proplist: typeof
    mdib_object src_proplist_ptr, // [in] assigned proplist: ptr to
)
```

5.4.4 Typed put value

Synopsis

```
int mdib_put_value_as_str8
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    str8 src_value // [in] src value
)

int mdib_put_value_as_str16
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    str16 src_value // [in] src value
)
```

```
int mdib_put_value_as_str32
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    str32 src_value // [in] src value
)

int mdib_put_value_as_int8
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    int8 src_value // [in] src value
)

int mdib_put_value_as_uint8
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    uint8 src_value // [in] src value
)

int mdib_put_value_as_int16
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    int16 src_value // [in] src value
)

int mdib_put_value_as_uint16
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    uint16 src_value // [in] src value
)

int mdib_put_value_as_int32
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    int32 src_value // [in] src value
)

int mdib_put_value_as_uint32
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    uint32 src_value // [in] src value
)

int mdib_put_value_as_int64
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    int64 src_value // [in] src value
)
```

```
int mdib_put_value_as_uint64
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier // [in] dst object name
    unit64 src_value // [in] src value
)

int mdib_put_value_as_int128
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    int128 src_value // [in] src value
)

int mdib_put_value_as_uint128
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    uint128_t src_value // [in] src value
)

int mdib_put_value_as_real32
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    float src_value // [in] src value
)

int mdib_put_value_as_real64
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    double src_value // [in] src value
)

int double mdib_put_value_as_real80
(
    mdib_handle session, // [in] session handle
    unsigned char *dst_identifier, // [in] dst object name
    long double src_value // [in] src value
)
```

5.5 Miscellaneous

5.5.1 Make Object message

Synopsis

```
integer mdib_make_object
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier, // [in] src object name
    unsigned char *src_label_type, // [in] assigned label: typeof
```

```
    mdib_object src_label_ptr, // [in] assigned label: ptr to
    unsigned char *src_type_type, // [in] assigned type: typeof
    mdib_object src_type_ptr, // [in] assigned type: ptr to
    unsigned char *src_object_type, // [in] assigned value: typeof
    mdib_object src_object_ptr, // [in] assigned value: ptr to
    unsigned char *src_proplist_type, // [in] assigned proplist: typeof
    mdib_object src_proplist_ptr, // [in] assigned proplist: ptr to
)
```

5.5.2 Remove Object message

Synopsis

```
integer mdib_remove_object
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier, // [in] src object name
)
```

5.5.3 Link Object message

Synopsis

```
integer mdib_link_object
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier, // [in] src object name
    unsigned char *dst_identifier, // [in] dst object name
    unsigned char *link_type // [in] link type: soft, hard
)
```

5.5.4 List Object message

Synopsis

```
string mdib_list_object
(
    mdib_handle session, // [in] session handle
    unsigned char *src_identifier // [in] src object name (wildcard)
)
```

6 Conformance

A conforming implementation shall conform to:

— the requirements of ISO/IEC 20944-03
— the requirements of ISO/IEC 20944-40
— the requirements of this Part