INTERNATIONAL STANDARD

ISO/IEC 20944-3

First edition
2013-01-15

Information technology — Metadata Registries Interoperability and Bindings (MDR-IB) —

Part 3: API bindings

Technologies de l'information — Interopérabilité et liaisons des registres de métadonnées (MDR-IB) —
Partie 3: Liaisons API

Reference number
ISO/IEC 20944-3:2013(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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

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

ISO/IEC 20944-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

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

— *Part 1: Framework, common vocabulary, and common provisions for conformance*

— *Part 2: Coding bindings*

— *Part 3: API bindings*

— *Part 4: Protocol bindings*

— *Part 5: Profiles*

Introduction

The ISO/IEC 20944 series of International Standards provides the bindings and their interoperability for metadata registries, such as those specified in the ISO/IEC 11179 series of International Standards.

This part of ISO/IEC 20944 contains provisions that are common to API bindings (Clauses 4-10) and the API bindings themselves (Clause 11 onward). The API bindings have commonality in their conceptualization of the services provided. For example, common features include:

— using a session paradigm to access data;

— using a parameterized security framework to support a variety of security techniques;

— using a hierarchical navigation for data access.

Clause 11 and onward are the actual API bindings themselves. The clauses of this part of ISO/IEC 20944 are organized such that future amendments are possible, which would include additional API bindings.

Information technology — Metadata Registries Interoperability and Bindings (MDR-IB) —

Part 3:
API bindings

1 Scope

The ISO/IEC 20944 series of International Standards describes codings, application programming interfaces (APIs), and protocols for interacting with an ISO/IEC 11179 metadata registry (MDR).

This part of ISO/IEC 20944 specifies provisions that are common across API bindings for the ISO/IEC 20944 series of International Standards, and provides the individual API bindings themselves.

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 9899:1999, *Programming languages — C*

ISO/IEC 11404:2007, *Information technology — General-Purpose Datatypes (GPD)*

ISO/IEC 13886:1996, *Information technology — Language-Independent Procedure Calling (LIPC)*

ISO/IEC 16262, *Information technology — Programming languages, their environments and system software interfaces — ECMAScript language specification*

ISO/IEC 20944-1:2013, *Information technology — Metadata Registries Interoperability and Bindings (MDR-IB) — Part 1: Framework, common vocabulary, and common provisions for conformance*

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 20944-1 apply.

4 Intended use of this part of ISO/IEC 20944

The purpose of this part of ISO/IEC 20944 is to provide a common set of services (common API provisions) and standardized API bindings such that portable programs can be written to access the MDR repositories. These programs are portable in the sense that the same program should behave similarly across all operating environments and the same program should be able to access MDR repositories that conform to this International Standard.

5 Abstract model

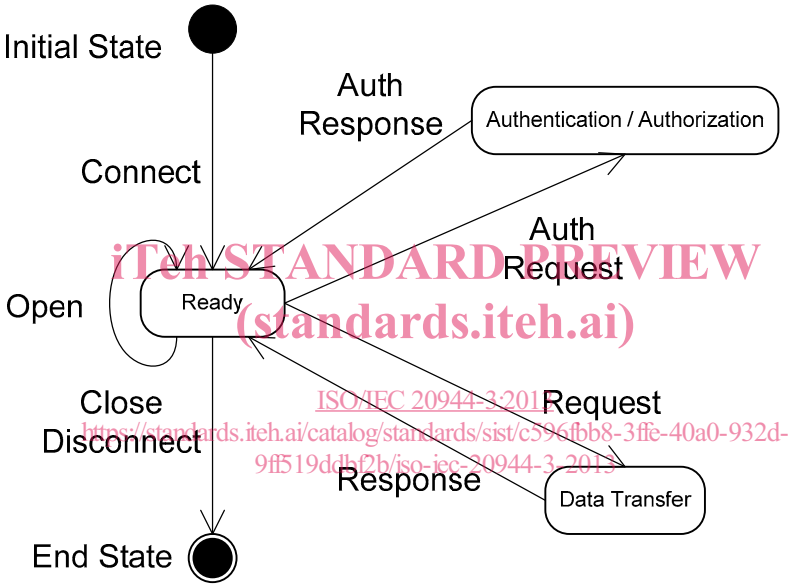
5.1 General

The API bindings have commonality in their conceptualization of data access services. For example, common features include:

— using a session paradigm to access data
— using a parameterized security framework to support a variety of security techniques
— using a hierarchical navigation for data access

5.2 Session paradigm

The following state diagram describes the different states of the services:



The states are:

— **initial state:** The initial state of an instance of the API.

— **end state:** The final state of an instance of the API.

— **ready:** The API is ready to for service requests.

— **authentication-authorization:** The API is processing an authentication-authorization request.

— **data transfer:** The API is processing a data transfer request.

The events are:

— **connect:** Setting up the initial connection.

— **open:** Adding more parameters to create a new handle and the context of a current node path for that handle.

— **close:** Destroying a handle created by open.

— **disconnect:** Knocking down a connection.

— **request:** A data transfer request.

— **response:** A data transfer response.

— **auth request:** An authentication-authorization request.

— **auth response:** An authentication-authorization response.

5.3 Security framework

The security framework provides a common technique for accessing security services and supports a common technique for implementing security services.

The common accessing technique uses the Request-Response Authorization-Authentication (RRAA) services. In the RRAA services, a Request is places for authorization, authentication, or some other security service. The Request provides the data, as required for the particular RRAA service. The RRAA service then provides a Response to the request. The services are symmetric in that both the API application (e.g., the program using the API) and the API environment (e.g., the underlying services and infrastructure) may each make requests of the other party, i.e., the API application can make requests to the API environment, and the API environment can make requests to the API application.

The supporting infrastructure should provide a run-time, dynamically configurable selection of RRAA services such that programs do not need to be recompiled or re-linked to take advantage of a new RRAA services or new security technologies[1].

This International Standard makes no requirements for a particular set of security services. The available services are implementation-defined.

5.4 Hierarchical navigation paths

The API services may access data via hierarchical navigation paths[2]. The navigation paths are based upon Uniform Resource Identifiers, as defined in RFC 3986. Absolute paths, which start at the top of a repository, are specified in subclause 3.3 of RFC 3986 under "absolute paths". Relative paths, which are relative to the current path[3], are specified in subclause 5 of RFC 3986 under "relative URIs".

The common accessing technique uses the Request-Response Authorization-Authentication (RRAA) services. In the RRAA services, a Request is places for authorization, authentication, or some other security service. The Request provides the data, as required for the particular RRAA service. The RRAA service then provides a Response to the request. The services are symmetric in that both the API application (e.g., the program using the API) and the API environment (e.g., the underlying services and infrastructure) may each make requests of the other party, i.e., the API application can make requests to the API environment, and the API environment can make requests to the API application.

---

1 See the white paper "Making Login Services Independent of Authentication Technologies", by Vipin Samar and Charlie Lai, that was presented at the Third ACM Conference on Computer Communications and Security (1996-03), which is available at

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.117.8767&rep=rep1&type=pdf

and the supporting API framework and services of Pluggable Authentication Modules (PAM) at

http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules

2 A hierarchical navigation path does not imply that the internal structure of data is hierarchical itself.

3 See Get Path and Put Path services in subclause 6.3.

6 Services

6.1 Use of ISO/IEC 13886

The notation of ISO/IEC 13886 Language-Independent Procedure Calls is used to describe service interfaces.

6.2 Session establishment services

The following services start up and shut down sessions with the metadata registries.

6.2.1 System information

Synopsis

```
mdrib_system_info:
procedure
(
    in session: mdrib_handle // session handle
)
returns characterstring,
```

Description

The **mdrib_system_info** service retrieves implementation-defined newline terminated, name-value pairs regarding the current configuration, limitations, and parameters of the session in **mdrib_handle**. If **mdrib_handle** is NULL, the information is for all sessions.

The following parameters are available on all implementations:

— "apistyle=xxx", where "xxx" is one of "synchronous" (API always blocks waiting for return), "nonblock" (API always returns immediately), "interrupt" (API interrupts when complete), "callback" (API performs callback when complete)

Example

```
// C/C++ illustration
printf(NULL, "mdrib configuiration:\n%s", mdrib_system_info());

// sample output
maximum_input_string=4095
supported_types=str8,int,char,long
version_info=3.17
version_description="C implementation with LDAP backend"
```

6.2.2 Configuration

Synopsis

```
// values accessible via "._system_info" identifier
data_object_identifier_length // Maximum supported data object identifier length
datatype_family_support // List of families supported
identifier_character_family_support // Allowed data object identifier characters
navigation_identifier_max // Maximum hierarchical navigation identifier length
session_max: Maximum number of simultaneous opened sessions
octet_transfer_max: // Maximum octets of data for a data object

// error codes accessible via "._error_family" and "._error_list"
error_family // Name of error code system
error_list // List of error codes and meanings
```

Description

The configuration features are accessible via Get and Put services via the identifiers "._system_info", "._error_family", and "._error_list".

Example

```
// C/C++ illustration
if ( strcmp(mdrib_get_value_as_str8(NULL,"._error_family",0,0,NULL),"POSIX") == 0 )
{
    // Error returns/handling are based upon POSIX standard
    // ...
}
```

6.2.3 Connect

Synopsis

```
mdrib_connect:
procedure
(
    in target: characterstring, // repository to connect to
    in options: characterstring, // connect options
)
returns mdrib_handle,
```


Description

Creates a new session to a data repository as named by target, an implementation-defined characterstring. The options parameter is a whitespace-separated list of name-value pairs that describe an implementation-defined set of connection options. Whitespace inside double-quoted characterstrings is escaped, i.e., this whitespace does not function as a name-value separator. If successful, returns a session handle to the repository, but does not request access (see mdrib_open).[4] If not successful, returns a null session handle.

Example

```
// C/C++ illustration
mdrib_handle sh; // session handle
mdrib_handle ch; // child session handle
sh = mdrib_connect("http://xyz.com/repository_2",
    "option_x=\"j k\" option_y=q option_z"); // note: option_x has the value "j k"
if ( sh != NULL )
{
    ch = mdrib_open(sh,"postal_address","read-only");
    // ...
    mdrib_close(ch);
    // ...
    mdrib_disconnect(sh);
}
```

---

[4] It is possible to combine connect, open, get, and close services in a single line of programming to give a state-less style of programming:

```
value = mdrib_get_as_str8( handle = mdrib_open( mdrib_connect("http://xyz.com/repository_2",
       "option_x=p option_y=q option_z"), "open options"), "city_name"), mdrib_close(handle);
```

6.2.4 Disconnect

Synopsis

```
mdrib_disconnect:
procedure
(
    in session: mdrib_handle // session handle
)
returns (state(success,failure)),
```

Description

Closes and disconnects a session to a data repository associated with the handle session and all of its child sessions. Returns success if successful, failure if unsuccessful.

Example

```
// C/C++ illustration
mdrib_handle sh; // session handle
sh = mdrib_connect("http://xyz.com/repository_2",
    "option_x=p option_y=q option_z");
// ... open session threads
// ... access metadata
// ... close session threads
mdrib_disconnect(sh);
```


6.2.5 Open

Synopsis

```
mdrib_open:
procedure
(
    in session: mdrib_handle, // session handle
    in node: characterstring, // portion of repository to open
    in options: characterstring, // open options
)
returns (mdrib_handle),
```

Description

Opens a child session within a session to data repository, as pointed to by the handle session.

The node parameter is the name of a portion of the repository — a view. If node is the empty string (""), then the child session is a duplicate session of the parent session. The partitioning and naming of contents of repositories is implementation-defined.

The options parameter is a whitespace-separated list of name-value pairs that describe a set of options from the following list:

— **"read-only"** (or **"ro"** or **"r"**): The child session is opened with read-only access.
— **"read-write"** (or **"rw"**): The child session is opened with read-write access.
— **"read-seq"** (or **"rs"**): The child session is opened with read sequential access. This option may be followed by the sub-option **"ordering=xxx"** where **"xxx"** may be **"breadth"**, **"depth"**, **"alphasort"**, or **"datesort"**.
— **"write-seq"** (or **"ws"**): The child session is opened with write sequential access.

— **"append"** (or **"a"**): The child session is opened for write-append access.
— **"inherit"** (or **""**): The child session inherits the characteristics of the parent session.

If mdrib_open is successful, it returns a handle to the child session. If not successful, it returns a null handle.

Example

```
// C/C++ illustration
mdrib_handle sh; // session handle
mdrib_handle ch; // child session handle
sh = mdrib_connect("http://xyz.com/repository_2",
    "option_x=p option_y=q option_z");
if ( sh != NULL )
{
    ch = mdrib_open(sh,"postal_address","read-only");
    // ...
    mdrib_close(ch);
    // ...
    mdrib_disconnect(sh);
}
```

6.2.6 Close

Synopsis

```
mdrib_close:
procedure
(
    in mdrib_handle: session, // session handle
)
returns (state(success,failure)),
```


Description

Closes a session associated with the handle session and all of its child sessions. Returns success if successful, failure if unsuccessful.

Example

```
// C/C++ illustration
mdrib_handle sh; // session handle
mdrib_handle ch; // child session handle
sh = mdrib_connect("http://xyz.com/repository_2",
    "option_x=p option_y=q option_z");
if ( sh != NULL )
{
    ch = mdrib_open(sh,"postal_address","read-only");
    // ...
    mdrib_close(ch);
    // ...
    mdrib_disconnect(sh);
}
```

6.3 Session parameter services

The following services may be used to modify and retrieve the parameters of the session.

6.3.1 Get path

Synopsis

```
mdrib_get_path:
procedure
(
    in session: mdrib_handle, // session handle
)
returns (characterstring),
```

Description

Retrieves the current node path for the session mdrib_handle.

If mdrib_get_path is successful, it returns a string containing the current node path; otherwise, error return is indicated by a return of a null pointer (not an empty string).

Example

```
// C/C++ illustration
mdrib_handle sh; // session handle
mdrib_handle ch; // child session handle
sh = mdrib_connect("http://xyz.com/repository_2",
    "option_x=p option_y=q option_z");
if ( sh != NULL )
{
    // open "postal_address"
    ch = mdrib_open(sh,"postal_address","read-only");
    // change path to "city"
    mdrib_put_path(ch,"city");
    // retrieve path, should be "city"
    new_path = mdrib_get_path(ch);
    if ( new_path != "city" )
    {
        // error
    }
}
```


6.3.2 Put path

Synopsis

```
mdrib_put_path:
procedure
(
    in session: mdrib_handle, // session handle
    in node: characterstring, // portion of repository
)
returns (state(success,failure)),
```

Description

Changes the current node path to the path specified by node for the session mdrib_handle. If node is a relative path, then the new path is relative to the previous node path.

If mdrib_put_path is successful, it returns success, otherwise error return is indicated by a return of failure.

Example

```
// C/C++ illustration
mdrib_handle sh; // session handle
mdrib_handle ch; // child session handle
sh = mdrib_connect("http://xyz.com/repository_2",
    "option_x=p option_y=q option_z");
if ( sh != NULL )
{
    // open "postal_address"
    ch = mdrib_open(sh,"postal_address","read-only");
    // change path to "city"
    if ( mdrib_put_path(ch,"city") == -1 )
    {
        // error
    }
    // success
}
```

6.4 Security services

The following services are supported.

6.4.1 Request Authorization/Authentication

Synopsis


```
mdrib_request_auth:
procedure
(
    in session: mdrib_handle, // session handle
    in auth_type: characterstring, // auth type
    in auth_options: characterstring, // auth options
)
returns (state(success,failure)),
```

Description

Requests the repository to supply authorization and/or authentication credentials, as pointed to by the handle session.

The auth_type parameter is a whitespace-separated list of name-value pairs that describe a set of credentials that are requested from the repository:

- **`"symmetric"`:** The authorization and/or authentication type is symmetric, i.e., both sides agree on the same **`"word"`**, such as a password. The auth_options parameter specifies a list of words to request as credentials.
- **`"asymmetric"`**: The authorization and/or authentication type is asymmetric, i.e., both sides agree on a separate set of **`"words"`**, such a public key techniques.
- **`"challenge"`**: Requests a response to the security challenge..
- **`"identifier"`**: Requests the repository supply a list of identifiers for authentication.
- **`"operation"`**: Requests the repository supply a list of operations to authorize.
- **`"nomad"`**: Requests agreement and authorization of a nomadic connection.