

Final Committee Draft ISO/IEC FCD 20944-4	
Date: 2008-05-27	Reference number: ISO/JTC 1/SC 32 N1757
Supersedes document SC 32N1572	

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: 2008-09-27 Please return all votes and comments in electronic form directly to the SC 32 Secretariat by the due date indicated.
---	---

ISO/IEC: FCD 20944-4:2008(E)
Title: Information technology - Metadata Registry Interoperability and Bindings (MDR-IB) Part 4: Protocol bindings
Project: 1.32.17.01.04.00

Introductory note: Text for FCD 20944-4; see CD1 disposition of comments N1758; this text is sent to NBs for 4 month letter ballot. The ballot starts 2008-05-27.
Medium: E
No. of pages: 23

Dr. Timothy Schoechle, Secretary, ISO/IEC JTC 1/SC 32
Farance Inc *, 3066 Sixth Street, Boulder, CO, United States of America
Telephone: +1 303-443-5490; E-mail: Timothy@Schoechle.org
available from the JTC 1/SC 32 WebSite <http://www.jtc1sc32.org/>
*Farance Inc. administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

Reference number of working document: **ISO/IEC JTC1 SC32 N1757**

Date: 2008-05-23

Reference number of document: **ISO/IEC FCD 20944-4**

Committee identification: **ISO/IEC JTC1 SC32 WG2**

SC32 Secretariat: **US**

**Information technology —
Metadata Registries Interoperability and Bindings (MDR-IB) —
Part 4: Protocol bindings**

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: **(40) Enquiry**

Document language: **E**

Copyright notice

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester:

*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*

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents

Page

Foreword	v
Introduction.....	vi
1 Scope	1
2 Normative references.....	1
3 Terms and definitions	2
4 Intended use of this Part.....	2
5 Abstract model	2
5.1 General	2
5.2 Participant model	2
5.3 Data model	2
5.4 Control Model	2
5.4.1 Event.....	3
5.4.2 Wait	3
5.4.3 Call	3
5.4.4 Parameters	3
5.5 Connections.....	3
5.6 State model	3
5.6.1 Server state model	3
5.6.2 Client state model.....	5
6 Services.....	6
6.1 Use of ISO/IEC 13886	6
6.2 Session establishment services	6
6.2.1 Connect	6
6.2.2 Disconnect	6
6.2.3 Open.....	7
6.2.4 Close.....	7
6.3 Session parameter services.....	7
6.3.1 Get path	7
6.3.2 Put path	8
6.4 Security services	8
6.4.1 Request Authorization/Authentication.....	8
6.4.2 Response Authorization/Authentication.....	9
6.5 Data transfer services	9
6.5.1 Get value	9
6.5.2 Put value.....	10
6.6 Miscellaneous	10
6.6.1 Make Object message.....	10
6.6.2 Remove Object message	11
6.6.3 Link Object message.....	11
6.6.4 List Object message.....	12
7 Bindings	12
8 Administration	12

9	Conformance	12
9.1	Protocol conformance paradigm	12
9.2	Protocol server service.....	13
9.3	Protocol client service	13
9.4	Conformance labels	13
10	Reserved for future standardization.....	13
11	HTTP/1.1 protocol binding.....	13
11.1	Session services	14
11.2	Service list.....	14
11.2.1	Get data service.....	14
11.2.2	Put data service	14
11.3	Conformance label prefix	14
12	WebDAV protocol binding	14
12.1	Session services	14
12.2	Service list.....	14
12.2.1	Get data service.....	15
12.2.2	Put data service	15
12.2.3	Get data property service	15
12.2.4	Put data property service	15
12.2.5	Remove data service.....	16
12.2.6	Make folder data service.....	16
12.3	Conformance label prefix	16

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 20944-4 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 Registries Interoperability and Bindings (MDRIB)*:

- *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

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

- common data transfer semantics
- harmonized session services for connection-oriented and connection-less protocols

Clause 11 and onward are the actual protocol bindings themselves. The clauses of this document are organized such that future amendments are possible, which would include additional protocol bindings.

Information technology — Metadata Registries Interoperability and Bindings (MDR-IB) — Part 4: Protocol bindings

1 Scope

The ISO/IEC 20944 family of standards describe codings, APIs, and protocols for interacting with an ISO/IEC 11179 metadata registry (MDR).

This Part of this International Standard specifies provisions that are common across protocol bindings for the 20944 family of standards, and this Part provides the individual protocol 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 11404:2007, *Information technology — General Purpose Datatypes (GPD)*

ISO/IEC 13886:1996¹, *Information technology — Language-Independent Procedure Calling (LIPC)*

ISO/IEC 20944-1:—², *Information technology — Metadata Registries Interoperability and Bindings (MDRIB) — Framework, common vocabulary, and common provisions for conformance*³

ISO/IEC 20944-2:—⁴, *Information technology — Metadata Registries Interoperability and Bindings (MDRIB) — Coding bindings*

IETF RFC 2068 (1997), *HTTP Hypertext Transfer Protocol — HTTP/1.1*

¹ ISO/IEC 13886 is freely available, see "<http://iso.org/ittf>" under Freely Available Standards.

² To be published.

³ The international standards, technical reports, and drafts of the 11179, 19773, 20943, and 20944 series are available at

<http://metadata-standards.org/11179>
<http://metadata-standards.org/19773>
<http://metadata-standards.org/20943>
<http://metadata-standards.org/20944>

⁴ To be published.

IETF RFC 2518 (1999), *HTTP Extensions for Distributed Authoring — WEBDAV*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in Part 1 apply⁵.

4 Intended use of this Part

The purpose of this Part is to provide a common set of services (common protocol provisions) and standardized protocol bindings such that interoperable systems can be created to access the MDR repositories. These systems are interoperable in they should be able to access MDR repositories that conform to this International Standard.

5 Abstract model

5.1 General

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

- common data transfer semantics
- harmonized session services for connection-oriented and connection-less protocols

The conceptual model is divided into two parts: the data model and the control model. The data model describes the logical structure of information as it is transferred. The control model describes the structure of the transactions, events, and protocol state.

5.2 Participant model

The senders and receivers of messages are called participants. The protocol binding permits client-server, peer-to-peer, publish-subscribe, multicast, and broadcast modes of participation. The underlying communication system determines which modes of participation are supported.

5.3 Data model

The data structure of 20944-2 is incorporated via normative reference.

5.4 Control Model

The control model refers to the methods of causing actions among the parties communicating. The following concepts characterize the control model.

⁵ Users and implementers of this International Standard may find it useful to reference additional terms and definitions from 20944-1.

5.4.1 Event

Assuming prior arrangement and appropriate authorization, each participant may send events. For the sender, an event is a one-way message with no expected response. In programming languages, an event is analogous to a "go to". For the receiver, an incoming event is processed by an event handler. The semantics of event handling is outside the scope of this Part.

5.4.2 Wait

Assuming prior arrangement and appropriate authorization, a participant can ask another participant to wait for an event, e.g., node A asks node B to wait for event E. In operating system kernel technology, this is equivalent to "sleeping on an event".

5.4.3 Call

Assuming prior arrangement and appropriate authorization, a participant can send a "call" message, similar to a remote procedure call. A "call" is equivalent to (1) sending data (input parameters) from caller to callee, (2) sending a "start" event to the "callee", (3) posting a "wait" on the callee's "finish" event, (4) receiving data (output parameters) from callee to caller. The advantage of "call" over combined "event" and "wait" is that "call" handles the storage of temporary information in the "event" and "wait" services.

5.4.4 Parameters

The "event", "wait", and "call" messages may pass parameters. Parameters are passed in a list along with the target event.

EXAMPLE A hypothetical function named `ls` is called with two function properties: (1) the options to the command, and (2) three parameters to the command (the third parameter has properties, too). The following illustration shows the syntax:

```
{ call ls .listing-type: long .dir-expand: no a b c .listing-encoding: iso-646 }
```

5.5 Connections

A connection is created when a client successfully completes a connection command to a server. A participant is one end of communication regardless of role (e.g., client or server). Some implementations may support Once a connection is established, parameters may be negotiated and sessions may be created for data and control transfer. A session is an connection established between one or more participants..

5.6 State model

A participant may act in the role of client or server.

5.6.1 Server state model

A server behaves according to the following state model.

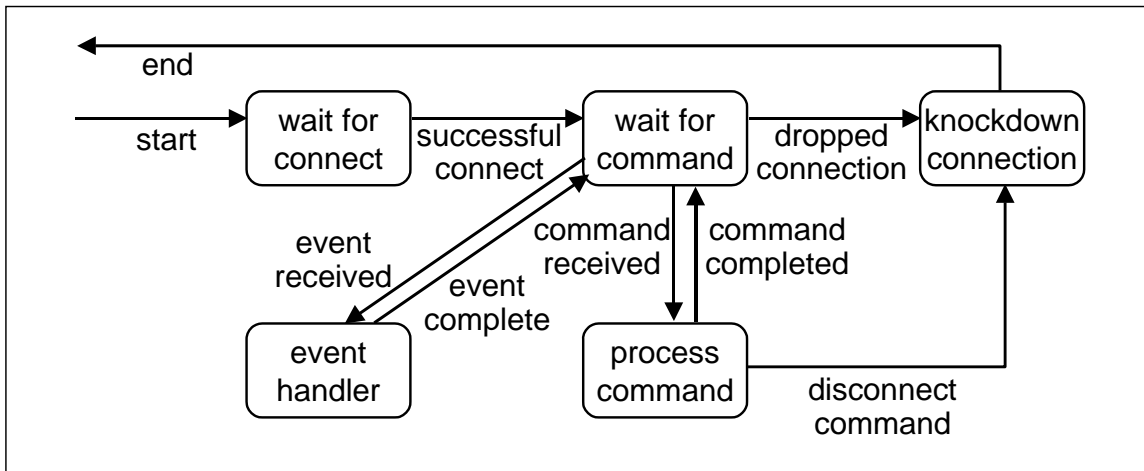


Figure 1. State model for server.

The following are the states for the server.

- Start: The state model begins in this state.
- Wait For Connect (state): The server is waiting for a connect.
- Successful Connect (event): Starts up the connection. Changes to Wait For Command (state).
- Wait For Command (state): The server is waiting for a command.
- Command Received (event): Begins the processing of a command. Changes to Process Command (state).
- Event Received (event): Begins the event handling. Changes to Event Handler (state).
- Dropped Connection (event): The connection was dropped. Changes to Knockdown Connection (state).
- Process Command (state): Processes a command, consumes and produces data, returns command status.
- Command Completed (event): The command has completed. Changes to Wait For Command (state).
- Disconnect Command (event): The Disconnect command was executed. Changes to Knockdown Connection (state).
- Event Handler (state): Processes an unsolicited event (e.g., security requests) and returns event status.
- Event Complete (event): The Event completed. Changes to Wait For Connect (state).
- Knockdown Connection (state): Closes the connection. Changes to End (state)
- End: The state model exits.

5.6.2 Client state model

A client behaves according to the following state model.

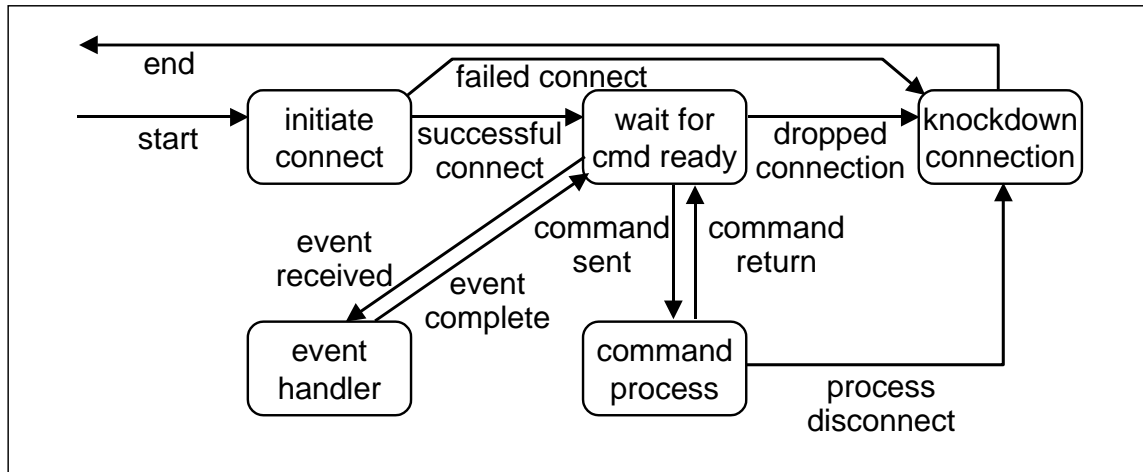


Figure 2. State model for client.

The following are the states for the client.

- Start: The state model begins in this state.
- Initiate Connect (state): The client initiates a connection the server.
- Successful Connect (event): Starts up the connection. Changes to Wait For Cmd Ready (state).
- Failed Connect (event): Connection failure. Changes to Knockdown Connection (state).
- Wait For Cmd Ready (state): The client is waiting for the server to indicate that it is ready for a command.
- Command Sent (event): Sends the command to process. Changes to Command Process (state).
- Event Received (event): Begins the event handling. Changes to Event Handler (state).
- Dropped Connection (event): The connection was dropped. Changes to Knockdown Connection (state).
- Command Process (state): Requests the processing of a command, produces and consumes data, receives the command status upon command completion.
- Command Return (event): The command has completed. Changes to Wait For Cmd Ready (state).
- Process Disconnect (event): Sends the Disconnect command. Changes to Knockdown Connection (state).
- Event Handler (state): Processes an unsolicited event (e.g., security requests) and returns event status.
- Event Complete (event): The Event completed. Changes to Wait For Cmd Ready (state).

- Knockdown Connection (state): Closes the connection. Changes to End (state)
- End: The state model exits.

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.

NOTE The use of 13886 notation is intended to provide a common description of services, but the protocol bindings themselves are not based upon APIs.

6.2 Session establishment services

The following messages start up and shut down sessions for protocols that use session-based access to metadata registries.

6.2.1 Connect

Synopsis

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

Description

Creates a new session to a data repository as named by target. The options parameter is an implementation-defined set of connection options. If successful, returns a session handle to the repository, but does not request access (see mdrib_open). If not successful, returns a null session handle.

6.2.2 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.

6.2.3 Open

Synopsis

```
mdrib_open:
procedure
(
    in session: mdrib_handle, // session handle
    in node: characterstring, // portion of repository to open
    in options: characterstring, // connect 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 outside the scope of this Standard.

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

6.2.4 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.

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 default node path.

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

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 default node path to the path specified by `node`. If `node` is a relative path, then the new path is relative to the previous default node path.

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

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 an implementation-defined set of credentials that are requested from the repository.

The `auth_options` parameter is an implementation-defined set of authorization and/or authentication options.

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

6.4.2 Response Authorization/Authentication

Synopsis

```
mdrib_response_auth:
procedure
(
    in session: mdrib_handle, // session handle
    in auth_type: string, // auth type
    auth_handler(): procedure, // auth handler function
)
returns state(success,failure),
```

Description

Registers a handler for authorization and/or authentication responses, as requested by the repository pointed to by the handle session.

The `auth_type` parameter is an implementation-defined set of credentials that are requested from the repository.

The `auth_handler` parameter a pointer to a handler service. The handler service is called for each authorization and/or authentication request that matches the type `auth_type`. The `auth_handler` service is called with at least three parameters: the session handle, the actual `auth_type`, the `auth_option`, and zero or more parameters.

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

6.5 Data transfer services

The following data transfer are supported.

6.5.1 Get value

Synopsis

```
mdrib_get_value:
procedure
(
    in session: mdrib_handle, // session handle
    in src_identifier: characterstring, // src object name
    in dst_label_type: characterstring, // saved label: typeof
    out dst_label_ptr: pointer, // saved label: ptr to
    in dst_type_type: characterstring, // saved type: typeof
    out dst_type_ptr: pointer, // saved type: ptr to
    in dst_object_type: characterstring, // saved value: typeof
```

```

        out dst_object_ptr: pointer, // saved value: ptr to
        in dst_proplist_type: characterstring, // saved proplist: typeof
        out dst_proplist_ptr: pointer, // saved proplist: ptr to
    )
    returns (state(success,failure)),

```

Description

Gets a value converted to a particular datatype, as identified by src_identifier.

6.5.2 Put value

Synopsis

```

integer mdrib_put_value
(
    in session: mdrib_handle, // session handle
    in src_identifier: characterstring, // src object name
    in src_label_type: characterstring, // assigned label: typeof
    in object src_label_ptr, // assigned label: ptr to
    in src_type_type: characterstring, // assigned type: typeof
    in object src_type_ptr, // assigned type: ptr to
    in src_object_type: characterstring, // assigned value: typeof
    in object src_object_ptr, // assigned value: ptr to
    in src_proplist_type: characterstring, // assigned proplist: typeof
    in object src_proplist_ptr, // assigned proplist: ptr to
)

```

Description

Puts a value in an object named src_identifier. The label is specified by the parameter type src_label_type and by a pointer to the parameter value src_label_ptr. The object type is specified by the parameter type src_type_type and by a pointer to the parameter value src_type_ptr. The object value is specified by the parameter type src_object_type and by a pointer to the parameter value src_object_ptr. The property list is specified by the parameter type src_proplist_type and by a pointer to the parameter value src_proplist_ptr. The parameter value may be MDRIB_PARAMETER_STRING that specifies the next parameter is a string type.

6.6 Miscellaneous

6.6.1 Make Object message

Synopsis

```

mdrib_make_object:
procedure
(
    in session: mdrib_handle, // session handle
    in src_identifier: characterstring, // src object name
    in src_label_type: characterstring, // assigned label: typeof
    in src_label_ptr: pointer, // assigned label: ptr to
    in src_type_type: characterstring, // assigned type: typeof
    in src_type_ptr: pointer, // assigned type: ptr to

```

```

    in src_object_type: characterstring, // assigned value: typeof
    in src_object_ptr: pointer, // assigned value: ptr to
    in src_proplist_type: characterstring, // assigned proplist: typeof
    in src_proplist_ptr: pointer, // assigned proplist: ptr to
)
returns (state(success,failure)),

```

Description

Creates a new object. The parameters are the same as the `mdrib_put_object`.

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

6.6.2 Remove Object message

Synopsis

```

mdrib_remove_object:
procedure
(
    in session: mdrib_handle, // session handle
    in src_identifier: characterstring, // src object name
)
returns (state(success,failure)),

```

Description

Removes an object. The `src_identifier` identifies the object to delete.

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

6.6.3 Link Object message

Synopsis

```

mdrib_link_object:
procedure
(
    session: mdrib_handle, // session handle
    in src_identifier: characterstring, // src object name
    in dst_identifier: characterstring, // dst object name
    in link_type: characterstring, // link type: soft, hard
)
returns (state(success,failure)),

```

Description

Links an existing object to a new name. The `src_identifier` is the name of the existing object. The `dst_identifier` is the name of the new object. The `link_type` is an implementation-defined link type.

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

6.6.4 List Object message

Synopsis

```
mdrib_list_object  
(  
    in session: mdrib_handle, // session handle  
    in src_identifier: characterstring, // src object name (wildcard)  
)  
returns (characterstring),
```

Description

Lists all objects that match src_identifier.

If mdrib_list_object is successful, it returns a string; otherwise, error return is indicated by the NULL pointer.

7 Bindings

This Part describes the common conceptual model and common data services across all protocol bindings. A conforming protocol binding shall conform to the requirements described in Clauses 4, 5, 6, 8, and 9 of this Part.

8 Administration

There are no administrative requirements.

9 Conformance

9.1 Protocol conformance paradigm

The conformance paradigm for 20944 consists of the following conformance roles: protocol client, protocol server, protocol peer.

Definitions: support, use, test, access, probe

The following terms are defined in the context of protocol conformance for data interchange participants:

- A "supported" feature is one that is implemented by the client, server, or peer service and may be used by any client, server, or peer service within the appropriate role among the data interchange participants, e.g., client-server, peer-to-peer.
- A feature is "used" if it is transmitted, received, or controlled by a client, server, or peer service.
- A feature is "tested" if client, server, or peer service inquires about or negotiates for the existence of that feature with other data interchange participants.
- A feature is "accessed" if it is transmitted or received by a client, server, or peer service.
- A feature is "probed" if a client, server, or peer service implicitly tests the existence of that feature by attempting to use the feature (see "use" above) with other data interchange participants that permit error recovery.

NOTE Protocol conformance makes requirements upon all data interchange participants: the client service, the server service, and the peer service.

9.2 Protocol server service

An strictly conforming protocol server shall implement all the features of the protocol according to the requirements of the protocol binding. A strictly conforming protocol server shall not implement extensions to the protocol binding.

An conforming protocol server shall implement all the features of the protocol according to the requirements of the protocol binding. A conforming protocol server may implement extensions to the protocol binding. A conforming protocol server may provide services to test, access, and/or probe for additional capabilities.

9.3 Protocol client service

An strictly conforming protocol client shall use the features of the protocol according to the requirements of the protocol binding. A strictly conforming protocol client shall not perform implementation-defined behavior.

An conforming protocol client shall use the features of the protocol according to the requirements of the protocol binding.

9.4 Conformance labels

The following labels indicate certain kinds of conformity to this Part:

- Pattern 1: "ISO/IEC 20944-4/B/prefix" where "prefix" represents the kind of binding, as defined in Clauses 11 and onward. For example, "ISO/IEC 20944-4/B/HTTP" corresponds to the HTTP/1.1 binding.
- Pattern 2: "ISO/IEC 20944-4/B/prefix/role" where "prefix" represents the kind of binding, as defined in Clauses 11 and onward, and "role" is one of "C", "S", or "P" corresponding to protocol client, protocol server, and protocol peer, as defined in this Clause. For example, "ISO/IEC 20944-4/B/WEBDAV/C" corresponds to a protocol client using the WebDAV protocol binding.

The location of the placement of conformance labels is outside the scope of this International Standard.

10 Reserved for future standardization

This Clause is reserved for future standardization.

11 HTTP/1.1 protocol binding

The HTTP/1.1 protocol binding is described in this Clause, based upon IETF RFC 2068 HTTP/1.1.

11.1 Session services

Within the context of MDR interoperability bindings, the HTTP/1.1 protocol binding is session-less⁶.

11.2 Service list

The HTTP/1.1 protocol binding uses the following services to affect data transfer:

11.2.1 Get data service

Data is transferred from server to client via the HTTP GET service.

The URI specified in the HTTP GET service corresponds to the path to the data itself.

EXAMPLE The following URL⁷

```
http://repository.org/VD;id=1234/value_domain_class/value_domain_unit_of_measure/unit_of_measure_precision
```

corresponds to the Unit of Measure Precision attribute for the Value Domain with item identifier "1234".

11.2.2 Put data service

This protocol binding does not provide a Put Data service.

11.3 Conformance label prefix

The prefix "HTTP" shall be used, in conjunction with 9.5, to indicate conformity to this Clause.

12 WebDAV protocol binding

The WebDAV protocol binding is described in this Clause, based upon IETF RFC 2518 WebDAV.

12.1 Session services

Within the context of MDR interoperability bindings, the WebDAV protocol binding is session-less⁸.

12.2 Service list

The WebDAV protocol binding uses the following services to affect data transfer:

⁶ The HTTP/1.1 protocol is session-based at Layer 5 of the protocol, but is session-less at Layer 7, which is what is described in this Part.

⁷ The syntax for repositories' URI is implementation-defined. The syntax in the illustration is hypothetical.

⁸ The WebDAV protocol is session-based at Layer 5 of the protocol, but is session-less at Layer 7, which is what is described in this Part.

12.2.1 Get data service

Data is transfer from server to client via the WebDAV GET service.

The URI specified in the WebDAV GET service corresponds to the path to the data itself. A collection of data may be retrieved, as per RFC 2518 subclause 8.4.

EXAMPLE The following URL corresponds to

```
http://repository.org/VD;id=1234/value_domain_class/value_domain_unit_of_measure/unit_of_measure_precision
```

corresponds to the Unit of Measure Precision attribute for the Value Domain with item identifier "1234".

12.2.2 Put data service

Data is transfer from server to client via the WebDAV PUT service.

The URI specified in the WebDAV PUT service corresponds to the path to the data itself. A collection of data may be stored, as per RFC 2518 subclause 8.7.

EXAMPLE The following URL corresponds to

```
http://repository.org/VD;id=1234/value_domain_class/value_domain_unit_of_measure/unit_of_measure_precision
```

corresponds to the Unit of Measure Precision attribute for the Value Domain with item identifier "1234". By using the PUT service and providing the data, the result is that the data is assigned to (replaces the value of) the Unit of Measure Precision attribute.

12.2.3 Get data property service

Data properties are retrieved via the WebDAV PROPFIND service.

The URI specified in the WebDAV PROPFIND service corresponds to the path to the data itself, as per RFC 2518 subclause 8.1.

EXAMPLE The following URL corresponds to

```
http://repository.org/VD;id=1234/value_domain_class/value_domain_unit_of_measure/unit_of_measure_precision
```

corresponds to the Unit of Measure Precision attribute for the Value Domain with item identifier "1234". The PROPFIND service retrieves properties associated with this data object.

12.2.4 Put data property service

Data properties are stored via the WebDAV PROPPATCH service.

The URI specified in the WebDAV PROPPATCH service corresponds to the path to the data itself, as per RFC 2518 subclause 8.2.

EXAMPLE The following URL corresponds to

```
http://repository.org/VD;id=1234/value_domain_class/value_domain_unit_of_measure/unit_
of_measure_precision
```

corresponds to the Unit of Measure Precision attribute for the Value Domain with item identifier "1234". The PROPFIND service stores properties associated with this data object.

12.2.5 Remove data service

Data is removed on the server via the WebDAV DELETE service.

The URI specified in the WebDAV DELETE service corresponds to the path to the data itself. A collection of data may be removed, as per RFC 2518 subclause 8.6.

EXAMPLE The following URL corresponds to

```
http://repository.org/VD;id=1234/value_domain_class
```

corresponds to the Value Domain with item identifier "1234". By using the DELETE service, the result is that the Value Domain is deleted.

12.2.6 Make folder data service

A data folder is created on the server via the WebDAV MKCOL service.

The URI specified in the WebDAV MKCOL service corresponds to the path to the folder itself, as per RFC 2518 subclause 8.3.

EXAMPLE The following URL corresponds to

```
http://repository.org/VD;id=1234;type=value_domain_class
```

corresponds to creating a Value Domain with item identifier "1234".

12.3 Conformance label prefix

The prefix "WEBDAV" shall be used, in conjunction with 9.5, to indicate conformity to this Clause.