

Australian Standard™

**Information technology—Multimedia
framework (MPEG-21)**

Part 10: Digital Item Processing



This Australian Standard was prepared by Committee IT-029, Coded Representation of Picture, Audio and Multimedia/Hypermedia Information. It was approved on behalf of the Council of Standards Australia on 19 April 2006.
This Standard was published on 8 May 2006.

The following are represented on Committee IT-029:

Australian Broadcasting Authority (ABA)
Australian Broadcasting Corporation (ABC)
Australian Consumers Association
Australian Subscription Television
CSIRO Mathematical & Information Services
Department of Defence (Australia)
Free TV Australia
School of Computer Science and Mathematics
Victoria University of Technology
Special Broadcasting Service (SBS)
The University of New South Wales
University of Wollongong

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about Standards can be found by visiting the Standards Web Shop at www.standards.com.au and looking up the relevant Standard in the on-line catalogue.

Alternatively, the printed Catalogue provides information current at 1 January each year, and the monthly magazine, *The Global Standard*, has a full listing of revisions and amendments published each month.

Australian Standards™ and other products and services developed by Standards Australia are published and distributed under contract by SAI Global, which operates the Standards Web Shop.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at mail@standards.org.au, or write to the Chief Executive, Standards Australia, GPO Box 476, Sydney, NSW 2001.

This Standard was issued in draft form for comment as DR 06074.

Australian Standard™

Information technology—Multimedia framework (MPEG-21)

Part 10: Digital Item Processing

First published as AS ISO/IEC 21000.10—2006.

COPYRIGHT

© Standards Australia

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Published by Standards Australia GPO Box 476, Sydney, NSW 2001, Australia

ISBN 0 7337 7396 6

PREFACE

This Standard was prepared by the Standards Australia Committee IT-029, Coded Representation of Picture, Audio and Multimedia/Hypermedia Information.

This Standard is identical with, and has been reproduced from ISO/IEC 21000-10:2006, *Information technology—Multimedia framework (MPEG-21)*, Part 10: *Digital Item Processing*.

The objective of this Standard is to provide the Australian multimedia industry with tools enabling users to provide suggested interactions with digital items to enable the inclusion of a dynamic aspect to static declarations of digital items.

Standards Australia is in the process of adopting most of ISO/IEC 21000 series Standards. Refer to the website for current list.

The terms ‘normative’ and ‘informative’ are used to define the application of the annex to which they apply. A normative annex is an integral part of a standard, whereas an informative annex is only for information and guidance.

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this part of ISO/IEC 21000’ should read ‘this Australian Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian Standard</i>	
ISO/IEC		AS ISO/IEC	
21000	Information technology – Multimedia framework (MPEG-21)	21000	Information technology – Multimedia framework (MPEG-21)
21000-2	Part 2: Digital Item Declaration	21000.2	Part 2: Digital Item Declaration
21000-3	Part 3: Digital Item Identification	21000.3	Part 3: Digital Item Identification
21000-5	Part 5: Rights Expression Language	21000.5	Part 5: Rights Expression Language
21000-6	Part 6: Rights data dictionary	21000.6	Part 6: Rights data dictionary
21000-7	Part 7: Digital item adaptation	21000.7	Part 7: Digital item adaptation
21000-9	Part 9: File Format	21000.9	Part 9: File Format

Only international references that have been adopted as Australian or Australian/New Zealand Standards have been listed.

CONTENTS

Page

1	Scope	1
2	Normative references	1
3	Terms, definitions, and abbreviated terms	2
4	Overview and Conventions.....	6
4.1	Overview of Digital Item Processing.....	6
4.2	Relation of Digital Item Processing with other parts of ISO/IEC 21000	7
4.3	Documentation conventions	7
4.4	Schema wrapper	8
4.5	Use of namespace prefixes	8
5	Digital Item Methods.....	9
5.1	Introduction	9
5.2	Digital Item Method Language.....	10
5.3	Digital Item Method linkage with DID.....	11
5.4	Digital Item Base Operations	21
5.5	Relation of Digital Item Base Operations and RDD verbs (informative)	56
5.6	Digital Item eXtension Operations	57
5.7	Auto run DIM	59
	Annex A (normative) ECMAScript binding for Digital Item Base Operations	62
	Annex B (normative) Java bindings for Digital Item Base Operations	64
	Annex C (normative) Calling MPEG-J based DIXOs from DIMs	74
	Annex D (informative) MPEG-J based model for execution of DIXOs	84
	Annex E (informative) XML Schema Definition for Digital Item Processing Elements	85
	Annex F (informative) A media handler implementation of play DIBO	87
	Annex G (informative) Tracking DIM execution for consistent rights checks	94
	Annex H (informative) Profiling DIP.....	97
	Annex I (informative) Digital Item Method Use Case Scenarios and Examples.....	101
	Bibliography	121

INTRODUCTION

Today, many elements exist to build an infrastructure for the delivery and consumption of multimedia content. There is, however, no “big picture” to describe how these elements, either in existence or under development, relate to each other. The aim for ISO/IEC 21000 (MPEG-21) is to describe how these various elements fit together. Where gaps exist, MPEG-21 will recommend which new standards are required. ISO/IEC JTC 1/SC 29/WG 11 (MPEG) will then develop new standards as appropriate while other relevant standards may be developed by other bodies. These specifications will be integrated into the multimedia framework through collaboration between MPEG and these bodies.

The result is an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. This open framework provides content creators and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumers, providing them access to a large variety of content in an interoperable manner.

The vision for MPEG-21 is to define a multimedia framework *to enable transparent and augmented use of multimedia resources across a wide range of networks and devices* used by different communities.

A key concept of the multimedia framework is the Digital Item. In MPEG-21 a Digital Item is a structured digital object with a standard representation, identification, and metadata. An equally important concept in the multimedia framework is the notion of the User. In MPEG-21 a User is any entity that interacts with the multimedia framework and as such includes all members of the value chain (e.g., creator, rights holders, distributors and consumers of Digital Items) and include, for example, individuals, consumers, communities, organizations, corporations, consortia, and governments.

Part 2 of MPEG-21 specifies the mechanism for declaring the structure and makeup of Digital Items. Such Digital Item Declarations are static by nature. This 10th part of MPEG-21 specifies tools enabling Users to provide suggested interactions with Digital Items, thereby enabling the inclusion of a dynamic aspect to the static declaration of Digital Items.

Information technology — Multimedia framework (MPEG-21) —

Part 10: Digital Item Processing

1 Scope

This Part of ISO/IEC 21000, entitled Digital Item Processing (DIP), specifies the syntax and semantics of tools that may be used to process Digital Items. The tools provide a normative set of tools that specify the processing of a Digital Item in a predefined manner.

This technology is specified in one normative clause and three normative annexes:

— Digital Item Methods:

Digital Item Methods (Clause 5) specifies the set of tools enabling Digital Item Users to include sequences of instructions for adding predefined functionality to a Digital Item. Such a sequence of instructions is a Digital Item Method. Digital Item Methods are authored with the Digital Item Method Language (see 5.2) which includes bindings to Digital Item Base Operations (see 5.4). For extended functionality, Digital Item eXtension Operations (see 5.6) allow such processing to be implemented more efficiently in a higher level programming language. Tools for integrating Digital Item Methods into Digital Item Declarations are also specified (see 5.3).

— ECMAScript bindings for Digital Item Base Operations:

Annex A specifies the ECMAScript bindings for the Digital Item Base Operations described in 5.3.

— Java bindings for Digital Item Base Operations:

Annex B specifies the Java bindings for the Digital Item Base Operations described in 5.4.

— Calling Java based DIXOs from Digital Item Methods:

Annex C specifies the mechanism for calling Java based Digital Item eXtension Operations. Digital Item eXtension Operations are described in 5.6.

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 16262:2002, *Information technology — ECMAScript language specification*

ISO/IEC 21000 (all parts), *Information technology — Multimedia framework (MPEG-21)*

IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, 1996

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

W3C REC-DOM-Level-3-Core-20040407, *Document Object Model (DOM) Level 3 Core Specification, Version 1.0*, W3C Recommendation 07 April 2004

W3C REC-DOM-Level-3-LS-20040407, *Document Object Model (DOM) Level 3 Load and Save Specification, Version 1.0*, W3C Recommendation 07 April 2004

W3C REC-xml-20040204, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation 04 February 2004

W3C REC-xml-names-19990114, *Namespaces in XML*, World Wide Web Consortium 14 January 1999

W3C REC-xmlschema-1-20041028, *XML Schema Part 1: Structures Second Edition*, W3C Recommendation 28 October 2004

W3C REC-xmlschema-2-20041028, *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation 28 October 2004

W3C REC-xpath-19991116, *XML Path Language (XPath), Version 1.0*, W3C Recommendation 16 November 1999

3 Terms, definitions, and abbreviated terms

For the purposes of this part of ISO/IEC 21000, the following terms, definitions and abbreviations apply:

3.1
Argument Type
type of the Digital Item Method Argument specified by an `Argument` element of the associated Digital Item Method declaration

NOTE 1 Argument Type is part of the Object Map allowing mapping of DIM Arguments to DID Objects.

NOTE 2 For further information see 5.3.5.

3.2
Digital Item
DI
structured digital object, including a standard representation, identification and metadata within the MPEG-21 framework

NOTE This entity is the fundamental unit of distribution and transaction within the multimedia framework as a whole.

[ISO/IEC TR 21000-1:2004, definition 2.3]

3.3
DIA
Digital Item Adaptation as specified by ISO/IEC 21000-7

3.4
Digital Item Base Operation
DIBO
base operation providing access to functionality implemented within an MPEG-21 environment and used in authoring a Digital Item Method

NOTE For further information see Clause 5.4.

3.5**Digital Item Base Operation implementation**

manner in which a particular implementer of a Digital Item Base Operation chooses to implement the normative semantics of the Digital Item Base Operation

3.6**Digital Item Declaration****DID**

declaration of the resources, metadata and their interrelationships of a Digital Item specified by ISO/IEC 21000-2

3.7**Digital Item Declaration Language****DIDL**

XML-based language including validation rules specified by ISO/IEC 21000-2 for the standard representation in XML of a Digital Item Declaration

3.8**Digital Item Declaration Language document**

a document using the Digital Item Declaration Language to declare a Digital Item in a standard representation in XML specified by ISO/IEC 21000-2

3.9**Digital Item Declaration Language element**

XML element of the Digital Item Declaration Language specified by ISO/IEC 21000-2

3.10**Digital Item Declaration Model**

set of abstract terms and concepts specified by ISO/IEC 21000-2 forming a model for declaring Digital Items

3.11**Digital Item Declaration Model entity**

entity of the Digital Item Declaration Model specified by ISO/IEC 21000-2

3.12**Digital Item Declaration Object****Object**

object representation in the Digital Item Method Language of a Digital Item Declaration element and associated with an Object Type

NOTE 1 A Digital Item Declaration Object has an Object Type that allows it to be processed in a Digital Item Method according to the Object Type. A Digital Item Declaration element is mapped to an Object Type by the Object Map.

NOTE 2 An Object Type is associated with a Digital Item Declaration element by an `ObjectType` element contained in a `DESCRIPTOR-STATEMENT` child of the Digital Item Declaration element. For further information see 5.3.

NOTE 3 The capitalized term Object is used in this part of ISO/IEC 21000 to mean a Digital Item Declaration Object. Other uses of the term object without an initial uppercase letter is used for an object as understood in the context of object-oriented programming.

3.13**Digital Item eXtension Operation****DIXO**

operation allowing extended functionality to be invoked from a Digital Item Method

NOTE For further information see Clause 5.6.

3.14**Digital Item eXtension Operation Language****DIXL**

programming language in which a Digital Item eXtension Operation is defined

3.15**DII**

Digital Item Identification as specified by ISO/IEC 21000-3

3.16**Digital Item Method****DIM**

tool for expressing the suggested interaction of a User with a Digital Item at the level of the Digital Item Declaration

NOTE 1 For further information see Clause 5.

NOTE 2 A Digital Item Method is composed of a Digital Item Method definition and its declaration.

3.17**Digital Item Method Argument**

argument to a Digital Item Method as represented in the Digital Item Method Language

3.18**Digital Item Method declaration**

declaration of the Digital Item Method as being part of a particular Digital Item

NOTE For further information see Clause 5.3.

3.19**Digital Item Method definition**

code written in the Digital Item Method Language that defines the Digital Item Method and that is either embedded inline with the Digital Item Method declaration or located separately and referenced from the Digital Item Method declaration

NOTE Whether the Digital Item Method definition is embedded inline or referenced from a separate location, it is the Digital Item Method definition itself that is the *resource* (in terms of the Digital Item Declaration Model).

3.20**Digital Item Method Language****DIML**

language providing the syntax and structure for authoring a Digital Item Method utilizing the Digital Item Base Operations

NOTE For further information see Clause 5.2

3.21**Digital Item Processing engine**

component within an MPEG-21 environment that supports ISO/IEC 21000-10 and is responsible for providing such supporting functionality (including execution of Digital Item Methods)

3.22**DOM**

Document Object Model (see W3C REC-DOM-Level-3-Core-20040407)

3.23**End User**

User taking the role of consumer, i.e. being at the end of a value or delivery chain

EXAMPLE A human consumer or an agent operating on behalf of a human consumer, etc.

[ISO/IEC TR 21000-1:2004, definition 2.4]

3.24**GUI**

Graphical User Interface

3.25**IPMP**

Intellectual Property Management and Protection as specified by ISO/IEC 21000-4

3.26**JPEG**

Joint Photographic Experts Group

3.27**MIME**

Multipurpose Internet Mail Extensions (see IETF RFC 2046)

3.28**MP3**

MPEG-1/2 layer 3 (audio coding)

3.29**MPEG**

Moving Picture Experts Group

3.30**Object Map**

map of Digital Item Declaration elements in a Digital Item Declaration to Digital Item Declaration Objects with their associated Object Types

EXAMPLE An object map might map several `ITEM` elements to an Object Type of “music track”, and another `ITEM` element to an Object Type of “album information”.

NOTE For further information see 5.3.5.

3.31**Object Type**

type of the Digital Item Declaration Object specified by an `ObjectType` descriptor of the associated `DIDL` element

NOTE 1 Object Type is part of the Object Map allowing mapping of `DID` Objects to `DIM` Arguments.

NOTE 2 For further information see Clause 5.3.5.

3.32**Peer**

device or application that compliantly processes a Digital Item

[ISO/IEC TR 21000-1:2004, definition 2.7]

3.33**RDD**

Rights Data Dictionary as specified by ISO/IEC 21000-6

3.34**REL**

Rights Expression Language as specified by ISO/IEC 21000-5

3.35**URI**

Uniform Resource Identifier (see IETF RFC 3986)

3.36**URL**

Uniform Resource Locator (see IETF RFC 3986)

3.37**User**

entity that interacts in the MPEG-21 environment or makes use of Digital Items

NOTE This includes all members of the value chain (e.g., creator, rights holders, distributors and consumers of Digital Items).

[ISO/IEC TR 21000-1:2004, definition 2.9]

3.38**W3C**

World Wide Web Consortium

3.39**XML**

Extensible Markup Language (see W3C REC-xml-20040204)

4 Overview and Conventions

4.1 Overview of Digital Item Processing

The Digital Item Declaration Language described in ISO/IEC 21000-2 is for creating a static declaration. Digital Item Processing assists processing of a Digital Item by providing tools allowing a User to add User specified functionality to a Digital Item Declaration. The standardization of Digital Item Processing enables interoperability at the processing level.

A key component of Digital Item Processing is the Digital Item Method (see 5). A Digital Item Method is the tool whereby a User (as defined in 21000-1) specifies suggested interactions with the Digital Item. As such, Digital Item Methods provide a way for a User to specify a selection of suggested procedures for processing a Digital Item at the level of the Digital Item itself.

EXAMPLE A Digital Item representing a music album can contain a Digital Item Method to add a new music track to the album. Such a Digital Item Method can be used to ensure that the new music track is added to the Digital Item while maintaining a suggested format for the Digital Item Declaration of such a music album Digital Item (i.e., elements added in the correct place in the Digital Item Declaration structure, correct Descriptors are included, etc.).

A Digital Item Method is expressed using the Digital Item Method Language (see 5.2) which includes a binding for Digital Item Base Operations (see 5.4). Digital Item eXtension Operations (see 5.6) provide a mechanism that allows the functionality provided by the standard set of Digital Item Base Operations to be extended.

Digital Item Methods, and the Digital Item Base Operations and Digital Item eXtension Operations called by them, can be considered as requests to the Digital Item Processing engine to process the Digital Item in some manner, or to execute some action.

The interface through which a User interacts with a Digital Item using Digital Item Processing is implementation dependent. Some implementations might support specification of aspects of the interface by metadata included in the Digital Item. Some possible scenarios follow.

- On receipt of a Digital Item Declaration, a list of Digital Item Methods (contained or referenced from within the DIDL document representing the Digital Item) can be presented to the User. The User can choose a Digital Item Method and then the Objects on which it operates. The Digital Item Processing engine then executes the chosen Digital Item Method with the chosen Objects as arguments;

- On receipt of a Digital Item Declaration, a list of Objects is presented based on the presence of Identifiers of the DII XML Namespace. The User chooses one or more of these Object(s). A list of Digital Item Methods that takes as arguments the (set of) Object(s) is then presented to the User. The User selects a Digital Item Method that is then executed by the Digital Item Processing engine.

4.2 Relation of Digital Item Processing with other parts of ISO/IEC 21000

Digital Item Processing is related to ISO/IEC 21000-2 by providing normative tools that enable functionality to be included in a Digital Item.

Implementations of DIBOs might have requirements or choices of implementation related to other parts of ISO/IEC 21000. ISO/IEC 21000-4, for instance, is expected to require that DIBO implementations accessing governed resources are required to check for permissions before doing so. In such cases, DIBO implementations would check for permissions and, in so doing, may take advantage of information compliant with ISO/IEC 21000-5 and ISO/IEC 21000-6. DIBO implementations may also make use of information compliant with ISO/IEC 21000-7, if appropriate to the DIBO semantics.

NOTE Annex G provides guidance on how to support DIP while maintaining a level of interaction with a Digital Item that is consistent with the available rights.

Overall processing of a Digital Item remains largely at the discretion of an application. Digital Items are intended to be used throughout the delivery chain, and thus different applications and different Users will perform different overall processing of a Digital Item. Digital Item Methods can be regarded as a 'menu' of User interaction possibilities. Digital Item Methods can then be utilized during processing of Digital Items to understand the Digital Item Method author's suggested manner of User interaction with a Digital Item. Different Digital Item Methods can be authored such that they provide different suggested interactions appropriate for different Users at various junctures in the delivery chain. This part of ISO/IEC 21000 specifies how to author Digital Item Methods and integrate them in a Digital Item Declaration. It does not specify how to restrict access to a Digital Item Method. This can be achieved, by utilizing other parts of ISO/IEC 21000 such as ISO/IEC 21000-4 and ISO/IEC 21000-5.

4.3 Documentation conventions

Literal machine-readable character sequences are shown in `fixed width font`.

References to DID Model entity names are shown in *italics*.

References to DIDL element names are shown in `FIXED WIDTH SMALL CAPS FONT`.

Normative syntax for DIP tools specified by XML Schema declarations and definitions are shown in this document using a separate font and background as follows.

EXAMPLE

```
<complexType name="ExampleType">
  <simpleContent>
    <extension base="string"/>
  </simpleContent>
</complexType>
```

XML Schema declarations and definitions as in the above example are to be considered fragments of a complete schema within an XML schema wrapper as described in 4.4.

Normative semantics for XML Schema declarations and definitions are set out in a table as follows.

EXAMPLE

Semantics of `ExampleType`:

Name	Definition
<code>ExampleType</code>	Example type semantics.

4.4 Schema wrapper

XML Schema declarations and definitions provided as XML fragments are to be understood as fragments of a complete schema and contained within an XML Schema `schema` element as follows.

```
<?xml version="1.0"?>
<!-- Digital Item Processing ISO/IEC 21000-10 -->
<schema
  targetNamespace="urn:mpeg:mpeg21:2005:01-DIP-NS"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:dip="urn:mpeg:mpeg21:2005:01-DIP-NS"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
</schema>
```

4.5 Use of namespace prefixes

Qualified Names are written with a namespace prefix followed by a colon followed by the local part of the Qualified Name as shown in the following example.

EXAMPLE `dip:ObjectType`

For clarity, consistent namespace prefixes as listed below are used in this part of ISO/IEC 21000.

Table 1 — Mapping of prefixes to namespaces

Prefix	Namespace
dia	urn:mpeg:mpeg21:2003:01-DIA-NS
didl	urn:mpeg:mpeg21:2002:02-DIDL-NS
dii	urn:mpeg:mpeg21:2002:01-DII-NS
dip	urn:mpeg:mpeg21:2005:01-DIP-NS
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
xi	http://www.w3.org/2001/XInclude

NOTE The prefixes `xml` and `xmlns` are normatively defined by *Namespaces in XML* (see W3C REC-xml-names-19990114). All other prefixes are not normative and are used by convention for consistency in this part of ISO/IEC 21000.

For informative examples provided as XML fragments without namespace declarations, the default namespace by convention in this part of ISO/IEC 21000 is defined as `urn:mpeg:mpeg21:2002:02-DIDL-NS` and the different prefixes are bound to the namespaces as listed above.

5 Digital Item Methods

5.1 Introduction

Digital Item Methods (DIMs) provide for a programmatic invocation of Digital Item Base Operations (DIBOs). Thereby a suggested interaction of the User with the Digital Item is provided. DIMs should thus be viewed from a User perspective; they are intended to be related to User interaction with a Digital Item.

DIMs are intended for working with parts of a DI at the DID level. DIMs are not intended to be utilized for implementing the processing of media resources themselves.

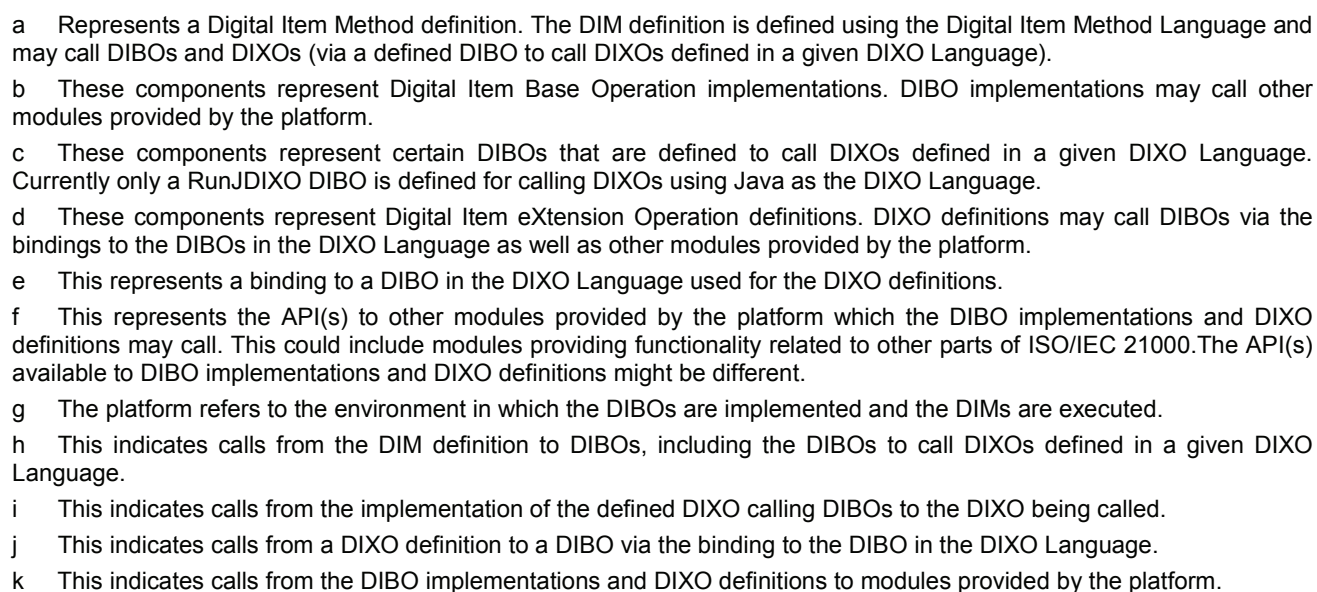
EXAMPLE DIMs are not intended to be used for implementing transcoding of media resources. However DIMs might be used for adaptations of the DID at the DID level.

NOTE While the intention is that media resource processing is not implemented directly within a DIM definition, DIBO implementations, such as the `adapt` DIBO (see 5.4.2.4.2), might have semantics that can lead to media resource processing. In addition media resource processing might take place in a DIXO (see 5.6).

Arguments to DIMs are DID Objects representing DIDL elements. The relationship between DIDL elements, DID Objects and DIM Arguments is specified by the Object Map (for further information on the Object Map see 5.3.5).

Digital Item Method tools include the following.

- Digital Item Base Operations (DIBOs) – the Digital Item Base Operations specify a high level normative interface to the basic types of interaction with a Digital Item. The set of normatively defined DIBOs have general application across a wide range of resources, applications, etc.;
- Digital Item Method Language (DIML) – the Digital Item Method Language specifies the normative language for defining interoperable DIMs and from which the DIBOs are able to be called. The ECMAScript binding of the DIBOs is a normative part of the DIML;
- Digital Item Method linkage with DID – this clause also specifies the normative mechanisms for including of Digital Item Methods in a DID;
- Digital Item Method execution – this clause also specifies the execution environment of a DIM; and
- Digital Item eXtension Operations (DIXO) – Digital Item eXtension Operations specify a normative mechanism for enabling functionality that extends beyond the basic functionality of the normative set of DIBOs in an efficient way.



5.2 Digital Item Method Language

The DIML specification normatively includes the ECMAScript Language Specification defined by ISO/IEC 16262:2002. This provides a standardized core language specification for DIML, including specification of the following features of DIML.

- AVA** Documents
Document Sharing Hub

- Composite data types (Objects and arrays);
- Standard arithmetic, logical and bitwise operators;
- Exception handling;
- Error definitions; and
- Support for Regular Expressions.

NOTE 1 ISO/IEC 16262:2002 is equivalent to ECMA-262 (edition 3).

NOTE 2 ISO/IEC 16262:2002 defines an **object** as an unordered collection of properties each of which contains a primitive value, object, or function. This part of ISO/IEC 21000 specifies DIML as an extension of ECMAScript by specifying additional objects. The DIBOs specified by this part of ISO/IEC 21000 are specified as function properties of these additional objects, hence in this part of ISO/IEC 21000 the term DIBO is used equivalently for the function properties of the objects specified by DIML.

5.2.2 Digital Item Base Operations

The DIML specification includes the normative set of DIBOs specified in 5.4.

5.2.3 Digital Item eXtension Operations

The DIML specification allows the calling of DIXOs as specified in 5.6.

5.3 Digital Item Method linkage with DID

5.3.1 Introduction

This clause specifies how DIMs are incorporated into a DID.

5.3.2 XML Schema for DIP

DIMs are incorporated into a DID using DIP defined XML (W3C REC-xml-20040204) elements that are incorporated at appropriate locations within the DIDL of the DID. W3C XML Schema (W3C REC-xmlschema-1-20041028, W3C REC-xmlschema-2-20041028) is used to define these DIP elements and the complete XML Schema for DIP is found in Annex E.

5.3.3 Digital Item Method declaration

5.3.3.1 Introduction

A Digital Item Method declaration is contained in a DIDL COMPONENT element which shall be constructed such that

- The COMPONENT should contain a DIM information descriptor represented by a DIP `MethodInfo` element contained in a DIDL DESCRIPTOR-STATEMENT. If the DIM does not have any arguments, the `MethodInfo` element need not be present;
- The COMPONENT may contain a flag, represented by a DIP `Label` element contained in a DIDL DESCRIPTOR-STATEMENT, indicating the DIM declaration is to be processed by the DIP engine; and
- The Digital Item Method definition is referenced or embedded by a RESOURCE child of the COMPONENT.

NOTE 1 A DIM declaration can be included from an external DIDL document by utilizing provisions for document modularity as specified in ISO/IEC 21000-2.

NOTE 2 An identifier can be associated with the DIM declaration using a DII Identifier (as specified by part 3 of ISO/IEC 21000). This could be located at the level of the COMPONENT, or at the level of the ITEM containing the COMPONENT (perhaps depending on the particular application of the DI). The RunDIM DIBO (see 5.4.2.7.11) supports both levels of identification.

NOTE 3 Since the `MethodInfo` element need not be present if the DIM does not have any arguments, it is not intended to be used to locate DIMs in a DID. For DIMs intended to be processed by the DIP engine, the `Label` element is used. In other cases the `contentType` attribute of the RESOURCE can be used.

5.3.3.2 MethodInfo syntax

```
<!--
#####
# Definition of MethodInfo                                     #
#####
-->
<element name="MethodInfo" type="dip:MethodInfoType"/>
<complexType name="MethodInfoType">
  <sequence>
    <element name="Argument" type="anyURI" minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
  <attribute name="autoRun" type="boolean" use="optional"
    default="false"/>
  <attribute name="profileCompliance"
    type="dip:ProfileComplianceType" use="optional"/>
</complexType>
<!--

#####
# Definition of ProfileComplianceType                         #
#####
-->
<simpleType name="ProfileComplianceType">
  <list itemType="QName"/>
</simpleType>
```

5.3.3.3 MethodInfo Semantics

Semantics of `MethodInfo`:

Name	Definition
MethodInfo	Content of this element is a sequence of <code>Argument</code> child elements. A DIDL COMPONENT representing a DIM declaration should contain a DIP <code>MethodInfo</code> element contained in a DIDL DESCRIPTOR-STATEMENT. There shall be one <code>Argument</code> child element for each argument of the DIM definition. The sequence of <code>Argument</code> child elements shall match the sequence of arguments of the DIM definition. If the DIM definition has no arguments and the <code>MethodInfo</code> element is still present, it shall contain zero <code>Argument</code> child elements.
Argument	Content of this element is a URI (IETF RFC 3986) indicating the Object Type associated with the corresponding argument of the DIM definition. There shall be a one-to-one mapping between the arguments of the DIM definition and a corresponding <code>Argument</code> element associating an Object Type with the

Name	Definition
	argument. The <code>Argument</code> elements shall be listed in the same sequence as the arguments of the DIM definition.
<code>autoRun</code>	This optional attribute allows the DIM author to identify an auto run DIM. If present and the value is true, then the declared DIM is an auto run DIM. For further information on auto run DIM see 5.7.
<code>profileCompliance</code>	<p>This optional attribute allows the DIM author to signal a profile or profiles to which the DIM conforms. If present, each member in the list is a Qualified Name representing one such profile. If present, this attribute need not provide a complete list of profiles. Processors may ignore this attribute. If they do process it, they may ignore any members of the list.</p> <p>NOTE Resolving the namespace prefix of the Qualified Name representing a profile to a namespace name URI reference is done using standard XML processing. That is, using an XML namespace declaration.</p>

5.3.3.4 Label syntax

```

<!--
#####
# Definition of Label                                     #
#####
-->
<element name="Label" type="dip:LabelType"/>
<complexType name="LabelType">
  <simpleContent>
    <extension base="anyURI"/>
  </simpleContent>
</complexType>

```

5.3.3.5 Label semantics

Semantics of `Label`:

Name	Definition
<code>Label</code>	<p>Content of this element is a URI (IETF RFC 3986). The presence of a <code>DIP Label</code> element contained in a <code>DIDL DESCRIPTOR-STATEMENT</code> indicates the parent element of the <code>DESCRIPTOR</code> is to be processed by the DIP engine. If present, such a <code>DESCRIPTOR</code> shall be the child of a <code>COMPONENT</code> representing a DIM declaration or DIXO declaration.</p> <p>For a DIM declaration that is to be processed by the DIP engine, the <code>Label</code> element shall be present and the value of the URI shall be <code>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</code>.</p> <p>For a DIXO declaration that is to be processed by the DIP engine, the <code>Label</code> element shall be present and the value of the URI shall be of the form <code>urn:mpeg:mpeg21:2005:01-DIP-NS:DIXO:x</code>, where <code>x</code> is unique for the DIXO language of the DIXO definition.</p>

Name	Definition
	<p>NOTE 1 The URI value for a <code>Label</code> of a J-DIXO is given in C.3.2.4.</p> <p>If a DIM declaration or DIXO declaration is not intended to be processed by the DIP engine, the <code>Label</code> element should be absent.</p> <p>NOTE 2 Processing by the DIP engine of a <code>COMPONENT</code> as a DIM declaration (or DIXO declaration) makes the DIM (or DIXO) available to the User for interacting with the DI (or calling from a DIM). If a DI is intended to be used, for example, to store or transport DIM declarations (and/or DIXO declarations), then these declarations can exclude a <code>Label</code>. When such a DI is processed, the declarations will not be processed by the DIP engine.</p>

5.3.3.6 Digital Item Method declaration examples

EXAMPLE The following example shows several different DIM declarations. In the example the DIM declarations are grouped together in a single DIDL `ITEM`, however this is not mandated by this part of ISO/IEC 21000.

The first DIM declaration provides an example of an empty `MethodInfo` element for a DIM that has no arguments. This DIM declaration also shows an example of an external reference to the DIM definition by referencing the location of the DIM definition using the `ref` attribute of the DIDL `RESOURCE` element. In addition, a second DIDL `RESOURCE` element referencing an equivalent DIM definition at an alternate location is provided.

The second DIM declaration provides an example of a `MethodInfo` element for a DIM that has two arguments. In this second DIM declaration, the DIM definition is embedded inline in the DIDL `RESOURCE` element (complete DIM definition not included for this example). This second DIM declaration also shows the use of a DIDL `CONDITION` element for conditional availability of this DIM declaration.

Both DIM declarations also show an example of including a plain text DIDL `DESCRIPTOR-STATEMENT` for containing a short human readable description of each DIM.

The third DIM declaration shows an example of signalling a DIM that is compliant with the `acme:basic` profile.

A fourth DIM declaration shows an example of using the provisions for document modularity specified in ISO/IEC 21000-2 to include elements of a DIM declaration contained in an external document.

Each of the DIM declarations discussed above contain a DIDL `DESCRIPTOR-STATEMENT` containing a DIP `Label`, indicating that these DIM declarations are to be processed by a DIP engine. Hence the DIMs declared by these DIM declarations will be available for User interaction with the DI containing this `ITEM`.

The DIM declaration in the external document `otherDI.xml` does not contain a DIP `Label`. In this example the `otherDI.xml` is being used only to store DIM declarations, and these DIM declarations are not intended to be processed by a DIP engine on processing of the `otherDI.xml` DI itself.

```

...
<Item id="DIMs">
  <Descriptor>
    <Statement mimeType="text/plain">DIM Declarations</Statement>
  </Descriptor>
  <Choice>
    <Selection select_id="selection1"/>
    <Selection select_id="selection2"/>
  </Choice>
  <Component id="dim_01">
    <Descriptor>
      <Statement mimeType="text/plain">Description of DIM 1</Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
      </Statement>
    </Descriptor>
  </Component>
</Item>

```

```

    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:MethodInfo/>
      </Statement>
    </Descriptor>
    <Resource mimeType="application/mp21-method"
      ref="http://www.somewhere.near/DIM#dim_01"/>
    <Resource mimeType="application/mp21-method"
      ref="http://www.somewhere.far/DIM#dim_01"/>
  </Component>
  <Component id="dim_02">
    <Condition require="selection1"/>
    <Descriptor>
      <Statement mimeType="text/plain">Description of DIM 2</Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:MethodInfo>
          <dip:Argument>urn:foo:type1</dip:Argument>
          <dip:Argument>urn:foo:type2</dip:Argument>
        </dip:MethodInfo>
      </Statement>
    </Descriptor>
    <Resource mimeType="application/mp21-method"><![CDATA[
      function dim_02( arg1, arg2 )
      {
        /* rest of DIM definition goes here */
      }
    ]]></Resource>
  </Component>
  <Component id="dim_03">
    <Condition require="selection2"/>
    <Descriptor>
      <Statement mimeType="text/plain">Description of DIM 3</Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:MethodInfo xmlns:acme="urn:acme:demo" profileCompliance="acme:basic">
          <dip:Argument>urn:foo:type3</dip:Argument>
        </dip:MethodInfo>
      </Statement>
    </Descriptor>
    <Resource mimeType="application/mp21-method"><![CDATA[
      function dim_03( arg1 )
      {
        /* rest of DIM definition goes here */
      }
    ]]></Resource>
  </Component>
  <Component>
    <xi:include href="otherDI.xml" xpointer="element(dim_04/1)"/>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
      </Statement>
    </Descriptor>
    <xi:include href="otherDI.xml" xpointer="element(dim_04/2)"/>
  </Component>

```

otherDI.xml:

5.3.4 Digital Item Method definition

A DIM definition is embedded in or referenced from a DIDL RESOURCE element. Such a RESOURCE element shall be contained in a DIM declaration as specified in 5.3.3.

The DIM definition itself is the sequence of operations authored using the DIML as specified in 5.2. This includes calling DIBOs, as specified in 5.4, to access the functionality available for User interactions with the DI provided by the implementation of the DIBO semantics.

A DIM definition may be embedded inline in a DID by base64 encoding the DIM definition prior to embedding or by placing the DIM definition in a CDATA section.

```
<Component id="dim_04">
  <Descriptor>
    <Statement mimeType="text/xml">
      <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
    </Statement>
  </Descriptor>
  <Resource mimeType="application/mp21-method" encoding="base64">
    cj5QUkVWMTk5MTQxMDkwNDQ5PC9BY2Nlc3Npb250dW1iZXI+PERhdGVQcm9kdWNlZD4xOTkxPC9E
    YXRlUHVJvZHVjZWQ+PE93bmVYPkNvcHlyaWdodCBCSU9TSVMgMjAwMzZwT3duZXI+PFByb2Rjc3N1
    ZVllYXI+MTk5MTwvUHJvZElzc3VlWWVhcj44QWJzdHJhY3RQcmVzZW50WU4+TjwvQWJzdHJhY3RQ
  </Resource>
</Component>
```

...

EXAMPLE 2 DIM definition embedded in a CDATA section

```

...
<Component id="dim_04">
  <Descriptor>
    <Statement mimeType="text/xml">
      <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
    </Statement>
  </Descriptor>
  <Resource mimeType="application/mp21-method"><![CDATA[
    function dim_04()
    {
      /* rest of DIM definition goes here */
    }
  ]]></Resource>
</Component>
...

```

5.3.4.3 Referenced

A DIM definition is referenced from a DIDL RESOURCE element by utilizing the `ref` attribute of the RESOURCE element. Referencing the DIM definition from the RESOURCE element allows the DIM definition to be located in a separate location from the DID (for example in a separate file).

EXAMPLE

```

...
<Component id="dim_05">
  <Descriptor>
    <Statement mimeType="text/xml">
      <dip:Label>urn:mpeg:mpeg21:2005:01-DIP-NS:DIM</dip:Label>
    </Statement>
  </Descriptor>
  <Descriptor>
    <Statement mimeType="text/xml">
      <dip:MethodInfo>
        <dip:Argument>urn:foo:bar</dip:Argument>
      </dip:MethodInfo>
    </Statement>
  </Descriptor>
  <Resource mimeType="application/mp21-method"
    ref="foop://some.where/dims/05"/>
</Component>
...

```

NOTE A file containing a DIM definition can contain more than one DIM definition provided that a URI can uniquely identify the required DIM definition.

5.3.5 Object Map

5.3.5.1 Introduction

An Object in a DID is associated with an Object Type using an Object Type descriptor. An Object Type descriptor is represented by a DIP `ObjectType` element contained in a DIDL `DESCRIPTOR-STATEMENT`. Any DIDL element that may contain a `DESCRIPTOR` may be associated with an Object Type. A single Object may be associated with zero or more Object Types. An Object Type descriptor is required for each Object Type associated with the Object.

An Object Type is associated with an argument of a DIM definition by a DIM arguments descriptor of the DIM declaration (see 5.3.3). An argument of a DIM shall be associated with only a single Object Type.

The Object Map is a map consisting of 2 parts (including their inter-relationship). The first part is a list of DID Objects (i.e., DIDL elements in a DIDL document that have a `DESCRIPTOR-STATEMENT` construct containing a DIP `ObjectType` element). The second part is a list of DIMs (that have zero or more DIM Arguments). When the value of an `ObjectType` element of a DID Object, and the value of the `Argument` element of a DIM are equal, then the DIM can be applied to that DID Object. Given a DIDL document, the Object Map describes these relationships between the DID Objects and the DIMs.

Thus the Object Map provides the link between the arguments of a DIM associated with a given Object Type, and the actual Objects in the DID that are associated with the same Object Type. An Object shall be allowed to be utilized as an argument to a DIM only if the Object is associated with the same Object Type as the argument.

Conceptually, an Object can be seen as a DIDL element that is mapped by the Object Map to an Object Type. The `MethodInfo` of the DIM declaration then describes what Object Type is associated with each argument of a DIM (see 5.3.3). This then determines which Objects (that is which DIDL elements that map to the required Object Type) may be used as actual arguments when the DIM is invoked.

EXAMPLE 1 In a DI representing a digital music album, one or more `ITEM` elements in the DID can be mapped to a “music track” Object Type. A “play track” DIM that plays a track of the music album can be authored such that it accepts an argument of Object Type “music track”. Any `ITEM` element assigned the Object Type “music track” can be utilized as an argument to the “play track” DIM.

EXAMPLE 2 The figure below depicts the Object Map of a DID, called “sample”, consisting of 3 Objects and 2 DIMs. The left hand side and right hand side of the figure depict two different views of this same DID. On the left hand side, each Object of the DID “sample” can be associated with an Object Type by using the DIP `ObjectType` element (see below) contained in a DIDL `DESCRIPTOR-STATEMENT`. The Object Type value of Object 1 and Object 2 is “urn:foo:bar:001”; Object 3 has Object Type “urn:foo:bar:002”. On the right hand side, the arguments to the DIMs contained in the DID “sample” are declared by a DIP `Argument` element (see 5.3.3). The Argument values of DIM 1 and DIM 2 are “urn:foo:bar:001” and “urn:foo:bar:002”, respectively. The centre of the figure depicts the Object Map which determines, via the Object Type, which Objects in the DID can be used as arguments to the DIMs in the DID. When the value of an `ObjectType` element of an Object, and the value of the `Argument` of a DIM are equal, then the DIM can be applied to that Object. From the left hand side view, Object 1 and Object 2 can be used as the Argument to DIM 1, while Object 3 can be used as the Argument to DIM 2. From the right hand side view, DIM 1 can be applied to Object 1 and Object 2; while DIM 2 can be applied to Object 3.

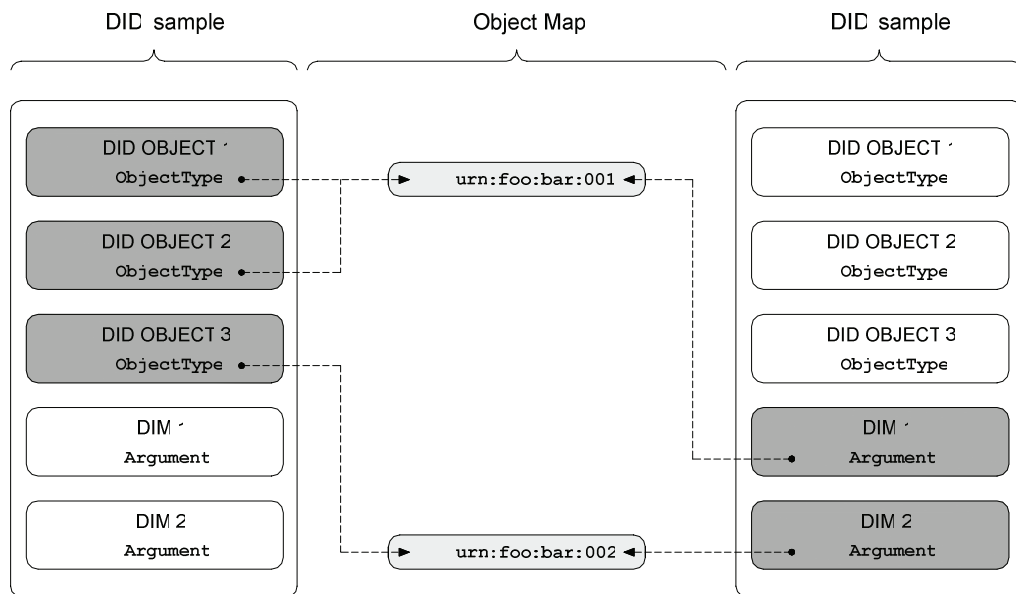


Figure 2 — Conceptual overview of Object Map

5.3.5.2 ObjectType syntax

```
<!--
#####
# Definition of ObjectType                                     #
#####
-->
<element name="ObjectType" type="dip:ObjectTypeType"/>
<complexType name="ObjectTypeType">
  <simpleContent>
    <extension base="anyURI"/>
  </simpleContent>
</complexType>
```

5.3.5.3 ObjectType Semantics

Semantics of `ObjectType`:

Name	Definition
<code>ObjectType</code>	Content of this element is a URI (IETF RFC 3986) indicating the Object Type associated with the parent DIDL element of the DIDL DESCRIPTOR containing the DIP <code>ObjectType</code> element.

5.3.5.4 ObjectType examples

The following example shows the use of `ObjectType` to associate several different DIDL elements with one or more Object Types.

EXAMPLE

```

...
<Item id="track1">
  <Descriptor>
    <Statement mimeType="text/xml">
      <dip:ObjectType>urn:foo:MusicTrack</dip:ObjectType>
    </Statement>
  </Descriptor>
  <Choice>
    <Selection select_id="audiobitrate_192k"/>
    <Selection select_id="audiobitrate_64k"/>
  </Choice>
  <Component id="track1audio_192k">
    <Condition require="audiobitrate_192k"/>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:ObjectType>urn:foo:Audio</dip:ObjectType>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:ObjectType>urn:foo:Audio192k</dip:ObjectType>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:ObjectType>urn:foo:AudioMP3</dip:ObjectType>
      </Statement>
    </Descriptor>
    <Resource mimeType="audio/mpeg" ref="mysong192kbps.mp3"/>
  </Component>
  <Component id="track1audio_64k">
    <Condition require="audiobitrate_64k"/>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:ObjectType>urn:foo:Audio</dip:ObjectType>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:ObjectType>urn:foo:Audio64k</dip:ObjectType>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:ObjectType>urn:foo:AudioMP3</dip:ObjectType>
      </Statement>
    </Descriptor>
    <Resource mimeType="audio/mpeg" ref="mysong64kbps.mp3"/>
  </Component>
  <Component id="track1score">
    <Descriptor>
      <Statement mimeType="text/xml">
        <dip:ObjectType>urn:foo:MusicScore</dip:ObjectType>
      </Statement>
    </Descriptor>
    <Resource mimeType="application/pdf" ref="mysongScore.pdf"/>
  </Component>
</Item>
...

```

5.4 Digital Item Base Operations

5.4.1 Introduction

The Digital Item Base Operations (DIBOs) are the functional building blocks utilized by a Digital Item Method (DIM). They can be considered somewhat analogous to the standard library of functions of a programming language.

Subclause 5.4 specifies the set of normatively defined DIBOs. The DIBOs are the functions (of the global and local objects) from which a DIM is built and they provide the interface between the DIM and the DIBO functionality provided by a DIP engine.

A DIBO is described by

- a normatively defined interface; and
- normatively defined semantics.

The interface specifies the DIBO name, the parameters of the DIBO, and whether the DIBO returns a value. The semantics of the DIBO specify the functionality of the DIBO, in addition to the semantics of the parameters and the return value, if any.

DIBO parameters may include

- DID Objects representing DIDL elements mapped to Object Types, and
- parameters bound to a type provided by the Digital Item Method Language described in 5.2.

NOTE 1 Types provided by DIML include standard ECMAScript types such as primitive values (e.g., Boolean, Number, or String values) and native objects (e.g., String, Number, Boolean and Array objects).

While a DIBO has a normative interface and normative semantics, this does not mean that DIBOs shall be implemented in a normative way. The DIBO implementation is left to the implementer of the DIBO. The DIBO implementation can thus be viewed as being the decision of the DIBO implementer as to how they have decided to provide the functionality defined by the DIBO semantics (with the functionality accessed from a DIM via the normative DIBO interface).

EXAMPLE The `play` DIBO (see 5.4.2.7.8) requests the playing of a *component* or *descriptor*, but implementers of the DIBO are able to play the specified DID Model entity in a manner of their choosing.

The DIBO implementer shall ensure that rights, if present are evaluated and enforced. This rights related functionality is not intended to be implemented within Digital Item Methods (by the DIM author), but instead shall be implemented by the DIBO implementer. Hence it is normative behaviour of the DIBO implementations that they execute according to the rights that are associated with the Digital Item under consideration.

Exceptions listed in the interface for each DIBO are those related specifically to the DIBO. Additional exceptions, such as runtime exceptions, may also be generated.

Most of the DIBOs described in this clause fall into one of the following categories.

- Operations that access and manipulate the DID at the DIDL level;

EXAMPLE DIBOs that add child nodes, remove child nodes, modify element attributes, etc. The interface and semantics for these operations are provided by including the DOM Level 3 Core API in DIML.

- Operations that load and serialize DID instance documents. The interface and semantics for these operations are provided by including the DOM Level 3 Load and Save API in DIML; and
- Operations related to particular parts of ISO/IEC 21000.

EXAMPLE DID related operations dealing with the state of DIDL CHOICE, SELECTION, and CONDITION elements. The interface and semantics for these operations are defined in following subclauses of this part of ISO/IEC 21000.

In addition Digital Item eXtension Operations, described in 5.6, enable extended functionality that cannot easily be realized in a DIM by making use of DIBOs in the above categories.

NOTE 2 The DIBOs defined in subclauses 5.4.2.4 to 5.4.2.8 are DIBO properties of the global objects specified in 5.4.2.1.

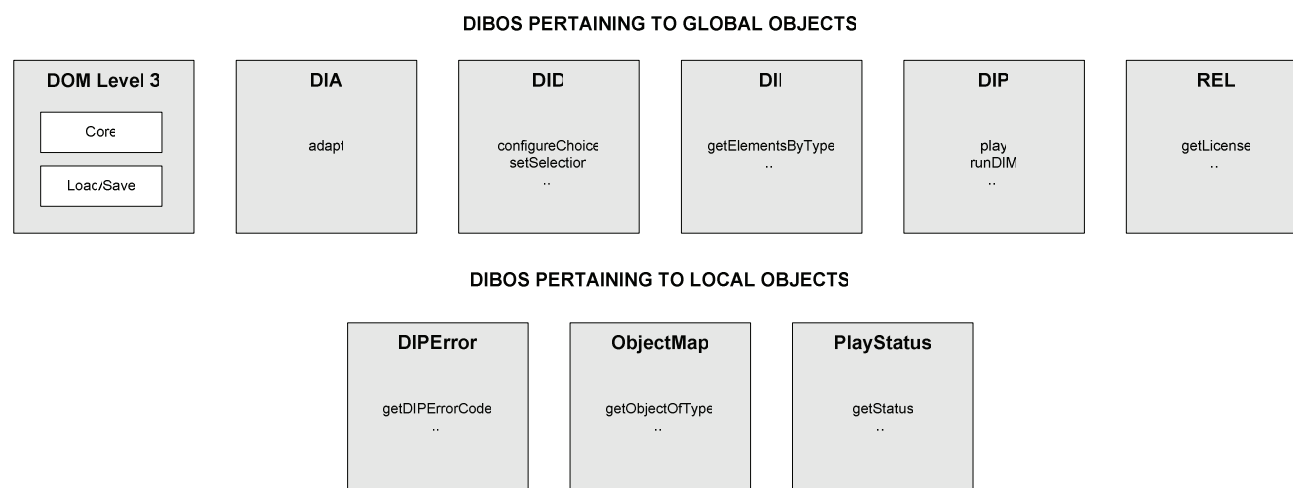


Figure 3 — Operations available to DIMs

5.4.2 Global objects, DIBOs and constants

5.4.2.1 Introduction

The DIML specification includes a normative set of DIP specific host-defined properties of the global object (see 15.1 of ISO/IEC 16262:2002). These DIP specific global properties include value properties and object properties, and are listed below. Unless otherwise stated these properties have the attributes { ReadOnly, DontDelete }.

Table 2 —DIML global values and objects

Value Properties:	MSG_INFO This global property is a number value (1) and may be used as a value for the <code>messageType</code> parameter of the <code>alert</code> DIBO to indicate the message is of a general informational nature. This property also has the attribute { DontEnum }.
	MSG_WARNING This global property is a number value (2) and may be used as a value for the <code>messageType</code> parameter of the <code>alert</code> DIBO to indicate the message is providing a warning. This property also has the attribute { DontEnum }.
	MSG_ERROR This global property is a number value (3) and may be used as a value for the <code>messageType</code> parameter of the <code>alert</code> DIBO to indicate the message is indicating some error condition has occurred. This property also has the attribute { DontEnum }.

	<p>MSG_PLAIN</p> <p>This global property is a number value (4) and may be used as a value for the <code>messageType</code> parameter of the <code>alert</code> DIBO to indicate the message is generic in nature. This property also has the attribute { DontEnum }.</p>
Object Properties:	<p>didDocument</p> <p>This global object property is a DOM <code>Document</code> object representing the current DID instance document. This is the DID containing (directly in the DID or by inclusion) the DIM declaration that is currently executing. In the case where a series of DIMs are in the execution stack (via one or more calls to the <code>RunDIM</code> DIBO) it is the DID containing the DIM declaration of the first DIM on the execution stack.</p> <p>NOTE This is analogous to the global <code>document</code> object specified in client-side JavaScript in web browsers.</p>
	<p>DIA</p> <p>This global property is an object that contains DIBOs providing DIA related functionality (see 5.4.2.4).</p>
	<p>DID</p> <p>This global object property is an object that contains DIBOs providing DID related functionality (see 5.4.2.5).</p>
	<p>DII</p> <p>This global property is an object that contains DIBOs providing DII related functionality (see 5.4.2.6).</p>
	<p>DIP</p> <p>This global object property is an object that contains DIBOs providing DIP related functionality (see 5.4.2.7).</p>
	<p>REL</p> <p>This global object property is an object that contains DIBOs providing REL related functionality (see 5.4.2.8).</p>

5.4.2.2 DIDL document access and manipulation

The hierarchical DID Model defined in part 2 of ISO/IEC 21000 reflects the structured nature of a Digital Item. This hierarchical DID Model is preserved by the hierarchical XML based representation (DIDL) of a DID also defined in part 2. The Document Object Model (DOM) defined by W3C is designed for the accessing and manipulation of such hierarchical documents.

Hence for Digital Item Methods the syntax and semantics of base operations for accessing and manipulating the DIDL document elements are those specified by the DOM Level 3 Core API as defined by W3C REC-DOM-Level-3-Core-20040407.

Since DIML is based on ECMAScript, the ECMAScript bindings of the DOM Level 3 Core API are included as part of the DIML specification.

5.4.2.3 DIDL document loading and saving

The syntax and semantics for operations for loading and saving a DIDL document are those specified by the DOM Level 3 Load and Save API as defined by W3C REC-DOM-Level-3-LS-20040407.

Since DIML is based on ECMAScript, the ECMAScript bindings of the DOM Level 3 Load and Save API are included as part of the DIML specification.

5.4.2.4 DIA related operations

5.4.2.4.1 Introduction

These DIBOs provide access to DIA related functionality. The DIBOs are provided as function properties of a `DIA` object. DIML includes such a `DIA` object as a property of the global object (see 5.4.2.1).

EXAMPLE Calling a DIA DIBO

```
function foo(arg1)
{
    ...
    DIA.adapt( component, metadata );
    ...
}
```

5.4.2.4.2 adapt

5.4.2.4.2.1 Interface

Syntax:	<code>adapt(element, metadata)</code>
Description:	Adapts a specified COMPONENT, or DESCRIPTOR element.
Parameters:	<p><code>element</code></p> <p>A DOM <code>Element</code> object representing the COMPONENT, or DESCRIPTOR element to be adapted.</p>
	<p><code>metadata</code></p> <p>Null or an array of DOM <code>Element</code> objects representing additional information that can be considered when adapting the element.</p>
Return value:	A DOM <code>Element</code> object representing the adapted DIDL element or null if the element was not adapted.
Exceptions:	<p><code>DIPError</code></p> <p>With DIP error code</p> <ul style="list-style-type: none"> — <code>ADAPTATION_FAILED</code> if an error occurs during adaptation of the resource; or — <code>INVALID_PARAMETER</code> if <ul style="list-style-type: none"> — <code>element</code> is not a COMPONENT, or DESCRIPTOR; or — or if <code>metadata</code> is not null nor an array of DOM <code>Element</code> objects.

5.4.2.4.2.2 Semantics

This DIBO allows a DIM author to explicitly request an adaptation of the DIDL element represented by the `element` parameter.

The `metadata` parameter is an array of `Element` objects representing additional information the DIM author suggests can be considered when adapting the resource. The `metadata` parameter may be null if the DIM author does not wish to provide any such suggestions. If the `metadata` parameter is not null and is not an array of DOM `Element` objects then an invalid parameter exception is generated.

If an adaptation of the element does take place and is successful, then an `Element` object representing the DIDL element for the adapted element is returned. This can then be utilized as a parameter to other appropriate DIBOs. The original element remains unchanged.

If an adaptation of the element is attempted but an error occurs during the adaptation, then a `DIPError` is generated.

If no adaptation of the element is attempted, then null is returned.

NOTE 1 An adaptation might not be attempted if, for example, adaptation is not supported, or the available metadata for adaptation is not supported.

NOTE 2 For an implementer of this DIBO, the available metadata to guide the adaptation can be sourced from the `metadata` parameter provided by the DIM author in the call to `adapt`, or else directly accessed from the DID from within the DIBO implementation. For example, if the DIBO is implemented in a Java environment that has standard DOM access to the DID, the DIBO implementer can directly inspect the `DESCRIPTORS`, if any, associated with the element for hints on adaptation. Also, metadata to guide the adaptation is not limited to the `metadata` parameter or metadata within the DID. For example metadata from a preferences or configuration file stored separately from the DID could be utilized, if the DIBO implementer has access to such information.

NOTE 3 The adaptation need not be implemented directly within the DIBO. For example, an implementer of this DIBO might choose to send a request along with relevant information to a remote server to do the adaptation.

NOTE 4 This DIBO allows the DIM author to explicitly request an adaptation. An adaptation of an element might still occur when the element is, for example, passed to the `play` DIBO. This would be a decision of the `play` DIBO implementer. In addition, such adaptations might still occur when elements are accessed without any interactions with the DI using DIP.

NOTE 5 ISO/IEC 21000-7 provides the following tools related to resource adaptation which an implementer of this DIBO can consider supporting.

- Usage Environment Descriptions;
- Bitstream Syntax Description based adaptation;
- Terminal and network quality of service;
- Usage Constraints Descriptions; and
- DIA Configuration.

5.4.2.4.2.3 Example use

EXAMPLE This DIBO would be required if a DIM author wanted to author a DIM that created a derived Digital Item containing a single resource created by adapting a resource from the original DI such that the resource in the derived DI is “pre-adapted” to the current environment.

5.4.2.5 DID related operations

5.4.2.5.1 Introduction

These DIBOs provide access to DID related functionality. The DIBOs are provided as function properties of a `DID` object. DIML includes such a `DID` object as a property of the global object (see 5.4.2.1).

EXAMPLE Calling a DID DIBO

```
function foo(arg1)
{
  ...
  choice = didDocument.getElementByID( "bitrate_choice" );
  DID. configureChoice( choice );
  ...
}
```

5.4.2.5.2 areConditionsSatisfied

5.4.2.5.2.1 Interface

Syntax:	<code>areConditionsSatisfied(element)</code>
Description:	Tests whether conditions for a specified DIDL element are satisfied.
Parameters:	<code>element</code> A DOM <code>Element</code> object representing the DIDL element for which conditions will be tested.
Return value:	A boolean value. A value of <code>true</code> is returned if the conditions are satisfied. A value of <code>false</code> is returned if the conditions are not satisfied or the state of the conditions cannot be resolved.
Exceptions:	<code>DIPError</code> With DIP error code <code>INVALID_PARAMETER</code> if <code>element</code> is not a DIDL element.

5.4.2.5.2.2 Semantics

This DIBO tests whether the conditions for a specified DIDL element are satisfied.

The `element` parameter specifies the DIDL element whose conditions are to be tested. It is an error to invoke this DIBO without specifying a DIDL element by the `element` parameter, in which case an invalid parameter exception is generated.

The return value is a `boolean` value indicating whether the conditions for the specified DIDL element are satisfied.

The conditions for the specified DIDL element are those described by any `CONDITION` child elements (as specified in ISO/IEC 21000-2) of the specified element. For the conditions of the specified element to be satisfied the rules for resolving `CONDITION` child elements as specified in ISO/IEC 21000-2 shall be evaluated.

EXAMPLE Multiple `CONDITION` child elements are evaluated in an “or” fashion based on the resolved state of each individual `CONDITION`.

If the specified `DIDL` element contains no child `CONDITION` elements, the conditions for the specified element are considered to be satisfied in context of this `DIBO` and a `true` value is returned.

In the context of this `DIBO`, the inability to resolve a `CONDITION` is interpreted as the condition being not satisfied. If all child `CONDITION` elements cannot be resolved or are false, a `false` value is returned.

If the specified `DIDL` element is a `CONDITION` element, then the predicates of the `CONDITION` are evaluated and the resulting value returned. If the `CONDITION` predicates cannot be resolved, `false` is returned.

NOTE The DIP engine usually will need to maintain its own internal state reflecting the state of the `CHOICES` and `SELECTIONS` of a `DID`. See also `configureChoice` (5.4.2.5.3) and `setSelection` (5.4.2.5.4).

5.4.2.5.2.3 Example usage

EXAMPLE An example where this `DIBO` can be used is at steps j), p), and u) in the scenario described in I.2.2.

5.4.2.5.3 configureChoice

5.4.2.5.3.1 Interface

Syntax:	<code>configureChoice(choice)</code>
Description:	Requests the User to configure a <code>CHOICE</code> element in the <code>DID</code> .
Parameters:	<code>choice</code> A <code>DOM Element</code> object representing the <code>CHOICE</code> element to be configured.
Return value:	Returns a boolean value. A value of <code>true</code> indicates the <code>CHOICE</code> configuration was modified, a value of <code>false</code> indicates the <code>CHOICE</code> configuration was not modified.
Exceptions:	<code>DIPError</code> With DIP error code <code>INVALID_PARAMETER</code> if <code>choice</code> is not a <code>CHOICE</code> .

5.4.2.5.3.2 Semantics

This `DIBO` requests the User to configure a specified `DIDL CHOICE` element.

The `choice` parameter specifies the `DIDL CHOICE` element to be configured by the User. It is an error to invoke this `DIBO` if the `choice` parameter does not specify a `CHOICE` element, in which case an invalid parameter exception is generated.

The `DIBO` allows the User to configure the `CHOICE` in compliance with ISO/IEC 21000-2 by making decisions about the *predicates* embodied by the `SELECTION` child elements of the `CHOICE`. If a `SELECTION` is chosen, its *predicate* becomes true, if it is rejected, its *predicate* becomes false, if it is unresolved, its *predicate* is undecided.

The `boolean` return value indicates whether the `CHOICE` configuration was modified or not.

NOTE 1 The manner of configuring the CHOICE is a DIBO implementation decision. For example, one DIBO implementation might present the CHOICE in a GUI to a human allowing the human to select or deselect SELECTION elements. Another implementation might be intended to process a DI without human intervention and implement some form of automated CHOICE configuration.

NOTE 2 The DIP engine will usually need to maintain its own internal state reflecting the state of the CHOICES and SELECTIONS of a DID. See also `setSelection` (5.4.2.5.4) and `areConditionsSatisfied` (5.4.2.5.2).

5.4.2.5.3.3 Example use

EXAMPLE An example where this DIBO can be used is at step m) in the scenario described in I.2.2

5.4.2.5.4 setSelection

5.4.2.5.4.1 Interface

Syntax:	<code>setSelection(selection, state)</code>
Description:	Sets the state of a given SELECTION element.
Parameters:	<p><code>selection</code></p> <p>A DOM <code>Element</code> object representing the SELECTION element whose state is to be set.</p>
	<p><code>state</code></p> <p>A string value specifying the state to set the SELECTION element. Valid values are <code>true</code>, <code>false</code>, or <code>undecided</code>.</p>
Return value:	None.
Exceptions:	<p><code>DIPError</code></p> <p>With DIP error code <code>INVALID_PARAMETER</code> if</p> <ul style="list-style-type: none"> — <code>selection</code> is not a SELECTION; or — <code>state</code> is not a string with value <code>true</code>, <code>false</code>, or <code>undecided</code>.

5.4.2.5.4.2 Semantics

This DIBO sets the state of a specified DIDL SELECTION element as maintained by the DIP engine.

The `selection` parameter specifies the DIDL SELECTION element whose state is to be set. It is an error to invoke this DIBO if the `selection` parameter does not specify a SELECTION element, in which case an invalid parameter exception is generated.

The `state` parameter specifies the value for the state of the SELECTION. This state is maintained by the DIP engine for subsequent reference in other DIBOs that rely on the configuration state of SELECTION elements (for example, `areConditionsSatisfied` 5.4.2.5.2). The valid values for the `state` parameter correspond to those SELECTION states described in ISO/IEC 21000-2, which are: `true`, `false`, `undecided`. It is an error to invoke this DIBO with any other value for the `state` parameter, in which case an invalid parameter exception is generated.

NOTE The DIP engine will usually need to maintain its own internal state reflecting the state of the CHOICES and SELECTIONS of a DID. See also `configureChoice` (5.4.2.5.3) and `areConditionsSatisfied` (5.4.2.5.2).

5.4.2.5.4.3 Example use

EXAMPLE An example where this DIBO can be used is when a DIM utilizes some input data to determine the state of a SELECTION, and then needs to call this DIBO to set the state of the SELECTION within the DIP engine execution environment.

5.4.2.6 DII related operations

5.4.2.6.1 Introduction

These DIBOs provide access to DII related functionality. The DIBOs are provided as function properties of a `DII` object. DIML includes such a `DII` object as a property of the global object (see 5.4.2.1).

EXAMPLE Calling a DII DIBO

```
function foo(arg1)
{
    ...
    elements = DII.getElementsByIdentifier( null, "urn:acme:foo:bar123" );
    ...
}
```

5.4.2.6.2 `getElementsByIdentifier`

5.4.2.6.2.1 Interface

Syntax:	<code>getElementsByIdentifier(sourceDID, value)</code>
Description:	Retrieves from the source DID existing DIDL elements that have a descriptor containing a DII Identifier with the specified value.
Parameters:	<p><code>sourceDID</code></p> <p>The DOM <code>Document</code> object representing the source DID from which the DIDL elements are to be retrieved.</p>
	<p><code>value</code></p> <p>A string value specifying the value of the DII Identifier identifying the DIDL elements to be retrieved.</p>
Return value:	An array of DOM <code>Element</code> objects representing the retrieved DIDL elements. May be empty if no DIDL elements matching the specified criteria are retrieved.
Exceptions:	<p><code>DIPError</code></p> <p>With DIP error code <code>INVALID_PARAMETER</code></p> <ul style="list-style-type: none"> — if <code>sourceDID</code> is not a DOM <code>Document</code> representing a DIDL instance document; or — <code>value</code> is not a string value.

5.4.2.6.2.2 Semantics

This DIBO retrieves from the specified DID existing DIDL elements based on a DII Identifier.

The `sourceDID` parameter is a DOM `Document` object representing the source DIDL document from which the elements are to be retrieved. It is an error to invoke this DIBO if the `sourceDID` parameter is not a DOM `Document` object, in which case an invalid parameter exception is generated.

The `value` parameter specifies the value of the DII Identifier required to identify the DIDL elements to be retrieved. The syntax and semantics of DII Identifier are specified in part 3 of ISO/IEC 21000. It is an error to invoke this DIBO if the `value` parameter is not a string value, in which case an invalid parameter exception is generated.

The retrieved elements are available to the invoking DIM via the returned array of `Element` objects. The elements can then be utilized as a parameter to other DIBOs. Only DIDL elements shall be returned. If no DIDL elements are identified by the specified criteria an empty array is returned.

5.4.2.6.2.3 Example use

EXAMPLE An example where this DIBO can be used is when a DIM author might not know where an element is located in the structure of a DID, but can determine a DII Identifier for the element. In this case the DIM author can utilize the `getElementsByIdentifier` DIBO to retrieve the element.

5.4.2.6.3 `getElementsByRelatedIdentifier`

5.4.2.6.3.1 Interface

Syntax:	<code>getElementsByRelatedIdentifier(sourceDID, value)</code>
Description:	Retrieves from the source DID existing DIDL elements that have a descriptor containing a DII RelatedIdentifier with the specified value.
Parameters:	<code>sourceDID</code> The DOM <code>Document</code> object representing the source DID from which the DIDL elements are to be retrieved.
	<code>value</code> A string value specifying the value of the DII RelatedIdentifier associated with the DIDL elements to be retrieved.
Return value:	An array of DOM <code>Element</code> objects representing the retrieved DIDL elements. May be empty if no DIDL elements matching the specified criteria are retrieved.
Exceptions:	<code>DIPError</code> With DIP error code <code>INVALID_PARAMETER</code> <ul style="list-style-type: none"> — if <code>sourceDID</code> is not a DOM <code>Document</code> representing a DIDL instance document; or — <code>value</code> is not a string value.

5.4.2.6.3.2 Semantics

This DIBO retrieves from the specified DID existing DIDL elements based on a DII RelatedIdentifier.

The `sourceDID` parameter is a DOM `Document` object representing the source DIDL document from which the elements are to be retrieved. It is an error to invoke this DIBO if the `sourceDID` parameter is not a DOM `Document` object, in which case an invalid parameter exception is generated.

The `value` parameter specifies the value of the DII RelatedIdentifier associated with the DIDL elements to be retrieved. The syntax and semantics of DII RelatedIdentifier are specified in part 3 of ISO/IEC 21000. It is an error to invoke this DIBO if the `value` parameter is not a string value, in which case an invalid parameter exception is generated.

The retrieved elements are available to the invoking DIM via the returned array of `Element` objects. The elements can then be utilized as a parameter to other DIBOs. Only DIDL elements shall be returned. If no DIDL elements are identified by the specified criteria an empty array is returned.

5.4.2.6.3.3 Example use

EXAMPLE This DIBO can be used when a DIM author has an identifier for some information and wants to retrieve any elements in the DID that identify that information as related information. For example, consider a Digital Item representing a collection of numerous musical pieces, with different sub-items for audio recordings, musical score, lyrics, and music videos, some of which might or might not be present for any individual musical piece. Given the identifier for a musical piece, the DIM author might want to retrieve in a single call all of the related sub-items. In this case the DIM author can utilize the `getElementsByRelatedIdentifier` DIBO to retrieve those elements (provided they utilize a DII RelatedIdentifier to identify the musical piece as being related).

5.4.2.6.4 getElementsByType

5.4.2.6.4.1 Interface

Syntax:	<code>getElementsByType(sourceDID, value)</code>
Description:	Retrieves from the source DID existing DIDL ITEM elements that have a descriptor containing a DII Type with the specified value.
Parameters:	<p><code>sourceDID</code></p> <p>The DOM <code>Document</code> object representing the source DID from which the DIDL elements are to be retrieved.</p>
	<p><code>value</code></p> <p>A string value specifying the value of the DII Type of the DIDL ITEM elements to be retrieved.</p>
Return value:	An array of DOM <code>Element</code> objects representing the retrieved DIDL ITEM elements. May be empty if no DIDL ITEM elements matching the specified criteria are retrieved.
Exceptions:	<p>DIPError</p> <p>With DIP error code <code>INVALID_PARAMETER</code></p> <ul style="list-style-type: none"> — if <code>sourceDID</code> is not a DOM <code>Document</code> representing a DIDL instance document; or — <code>value</code> is not a string value.

5.4.2.6.4.2 Semantics

This DIBO retrieves from the specified DID existing DIDL elements based on a DII Type.

The `sourceDID` parameter is a DOM `Document` object representing the source DIDL document from which the elements are to be retrieved. It is an error to invoke this DIBO if the `sourceDID` parameter is not a DOM `Document` object, in which case an invalid parameter exception is generated.

The `value` parameter specifies the value indicating the DII Type of the DIDL ITEM elements to be retrieved. The syntax and semantics of DII Type are specified in part 3 of ISO/IEC 21000. It is an error to invoke this DIBO if the `value` parameter is not a string value, in which case an invalid parameter exception is generated.

The retrieved elements are available to the invoking DIM via the returned array of `Element` objects. The elements can then be utilized as a parameter to other DIBOs. Only DIDL ITEM elements shall be returned. If no DIDL ITEM elements are identified by the specified criteria an empty array is returned.

5.4.2.6.4.3 Example use

EXAMPLE An example where this DIBO can be used is when Digital Item is used as a container of several sub-items of different DII Types. In this case the DIM author can utilize the `getElementsByType` DIBO to retrieve all ITEM elements representing sub-items of a given DII Type.

5.4.2.7 DIP related operations

5.4.2.7.1 Introduction

These DIBOs provide access to DIP related functionality. The DIBOs are provided as function properties of a `DIP` object. DIML includes such a `DIP` object as a property of the global object (see 5.4.2.1).

EXAMPLE Calling a DIP DIBO

```
function foo(arg1)
{
  ...
  DIP.Play( component, false );
  ...
}
```

5.4.2.7.2 alert

5.4.2.7.2.1 Interface

Syntax:	<code>alert(message, messageType)</code>
Description:	Alerts the User with a message.
Parameters:	<p><code>message</code></p> <p>A string value containing the message to be displayed.</p>
	<p><code>messageType</code></p> <p>A number value indicating the generic nature of the message.</p> <p>Valid values are <code>MSG_INFO</code>, <code>MSG_WARNING</code>, <code>MSG_ERROR</code> and <code>MSG_PLAIN</code> which are defined as value properties of the DIML global object (see 5.4.2.1).</p>

Return value:	None
Exceptions:	<p>DIPErrors</p> <p>With DIP error code <code>INVALID_PARAM</code></p> <ul style="list-style-type: none"> — If <code>message</code> is not a string value; or — if <code>messageType</code> does not specify a valid value.

5.4.2.7.2.2 Semantics

This DIBO provides simple textual feedback to the User.

EXAMPLE To notify the User of an error condition.

The textual message is specified by the `message` parameter. It is an error to invoke this DIBO if the `message` parameter is not a string value, in which case an invalid parameter exception is generated.

The `messageType` parameter indicates the nature of the message. It is an error to invoke this DIBO if the `messageType` parameter is not a valid number value, in which case an invalid parameter exception is generated (a valid number value is one of the values `MSG_INFO`, `MSG_WARNING`, `MSG_ERROR`, or `MSG_PLAIN` defined as number properties of the DIML global object).

NOTE The implementation of the alert is a DIBO implementation decision. For example, one DIBO implementation might present the message in a GUI to a human user. Another implementation might be intended to process a DI without human intervention and log the message to a log file.

5.4.2.7.2.3 Example use

EXAMPLE An example where this DIBO can be used is at step z) in the scenario described in I.2.2.

5.4.2.7.3 execute

5.4.2.7.3.1 Interface

Syntax:	<code>execute(element)</code>
Description:	Executes the resource associated with a specified COMPONENT, or DESCRIPTOR.
Parameters:	<p><code>element</code></p> <p>A DOM <code>Element</code> object representing the COMPONENT, or DESCRIPTOR element containing the executable resource.</p>
Return value:	A boolean value of <code>true</code> if the resource execution was successfully initiated, or <code>false</code> if resource execution was not initiated.

Exceptions:	<p><code>DIPError</code></p> <p>With DIP error code</p> <ul style="list-style-type: none"> — <code>INVALID_PARAMETER</code> if <ul style="list-style-type: none"> — <code>element</code> is not an <code>COMPONENT</code>, or <code>DESCRIPTOR</code>; or — more than one executable digital resource is found to be associated with the specified element; or — <code>EXECUTE_FAILED</code> if an error occurs during the execution.
-------------	--

5.4.2.7.3.2 Semantics

This DIBO causes execution to be initiated for the executable resource associated with the DIDL element represented by the `element` parameter.

The `element` parameter shall be a DOM `Element` object representing a `COMPONENT` or `DESCRIPTOR` containing the executable digital resource to be executed. It is an error to invoke this DIBO if the `element` parameter is not a DOM `Element` object representing a `COMPONENT` or `DESCRIPTOR`, in which case an invalid parameter exception is generated.

The manner of executing the associated resource, appropriate to its media type, is left as an implementation choice of the DIBO implementer.

NOTE 1 It is the responsibility of the DIBO implementation and DIP engine to ensure security related issues are addressed when executing executable resources.

NOTE 2 When determining the associated executable digital resource, the case where there is a `COMPONENT` with multiple `RESOURCE` child elements still represents a single *resource*.

5.4.2.7.3.3 Example use

EXAMPLE An example where this DIBO can be used is when a DI author might want to include as a resource in a Digital Item an executable program that is capable of presenting other resources in the Digital Item that are in a proprietary format understood only by the executable resource and that utilizes descriptors attached to the resources containing proprietary metadata to determine the presentation.

5.4.2.7.4 `getExternalData`

5.4.2.7.4.1 Interface

Syntax:	<code>getExternalData(mimeTypes, requestMessages)</code>
Description:	Requests the User to select resources located external to the DI.
Parameters:	<p><code>mimeTypes</code></p> <p>An array of arrays of string values specifying the media formats required.</p>
	<p><code>requestMessages</code></p> <p>An array whose elements are either string values or <code>null</code>.</p>