INTERNATIONAL STANDARD

ISO/IEC 24800-3

First edition
2010-05-01

AMENDMENT 1
2015-12-15

Information technology — JPSearch —

Part 3: Query format

AMENDMENT 1: JPSearch API

Technologies de l'information — JPSearch —

Partie 3: Format d'interrogation

AMENDEMENT 1: API JPSearch

Reference number
ISO/IEC 24800-3:2010/Amd.1:2015(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — JPSearch —

Part 3:
Query format

AMENDMENT 1: JPSearch API

Add the following paragraph at the end of Clause 1.

In addition, this part of ISO/IEC 24800 specifies the JPSearch API, a Restful State (REST) Application Programming Interface, complementary to the JPSearch Query Format (JPQF). It is primarily intended to be used for web services. Compared to JPQF, it does not only specify the query syntax, but also the protocols used for sending and retrieving queries. It reduces the complexity of expressing image search queries by providing a simplified syntax to express common queries in the form of URIs. Furthermore, the API allows embedding JPQF queries to express more advanced queries or to use the API solely as a communication protocol for sending and retrieving queries between client and repository. In addition to text based search queries, the API provides functionality to support visual search applications, such as content-based image retrieval systems.

Change the title of Clause 5 “Disabled datatypes” to “JPQF Disabled datatypes”.

Change the title of Clause 6 “Disabled Query Types” to “JPQF Disabled Query Types”.

Move Clause 7 “Conformance” to the end of the document and update numbering.

Add the following clauses to the end of the document (before Conformance).

7 Basic concepts of the JPSearch API

The API builds upon the HyperText Transfer Protocol (HTTP) for client-server communication. More specifically, in the context of a JPSearch-environment, the communication between a JPSearch-client and a JPSearch-server is specified. A JPSearch-client is any application that can formulate JPSearch-requests and handle JPSearch-responses. A JPSearch-server is an image repository that answers to JPSearch-requests and responds compliant with this part of ISO/IEC 24800.

Requests are largely expressed using the Uniform Resource Identifier (URI) syntax. This part of ISO/IEC 24800 follows the RFC 3986 syntax, as shown below:

```
URI = scheme ":" authority "/" path [ "?" query ] [ "#" fragment ]
```

More specifically, this part of ISO/IEC 24800 focuses solely on the “query” part of the former definition. The query part is a sequence of key value pairs separated with an “&” character. The keys are referred to as arguments or query options. For example, the following query has three query options arg1, arg2 and arg3:

```
?arg1=value1&arg2=value2&arg3=value3
```

Please note that the URIs should adopt proper URL encoding according to the RFC 3986 specification. Examples provided in this part of ISO/IEC 24800 are not encoded for the sake of readability.

Query options specified in this part of ISO/IEC 24800 are referred to as “system query options”. These query options can have the following types:

— signed or unsigned integer;

— real numbers;

— intervals: [from-to], for example, [0-10] denotes any integer number between 0 and 10, [0.-10.] denotes any real number between 0 and 10;

— boolean: 0 = false and 1 = true;

— string: strings should be URL encoded. Strings can be restricted to a specific syntax, e.g. a pair is a string of two values separated with a comma: “x,y”;

— enum: discrete list of valid values, where the options are separated with a “|”. For example: value1|value2|...;

— time: formatted string according to ISO 8601 representation;

— geolocation: formatted string containing two comma separated real numbers representing latitude and longitude as decimal degrees with negative numbers for south and west.

A JPSearch repository is completely free to specify the remaining part of the URI, i.e. the scheme, authority, path and fragments. Furthermore, additional query options can be specified, as long as they do not infer with the system query options. By default, JPSearch system query options do not have a prefix. However, in order to avoid collisions with custom query options, a prefix can be specified in the capability description.

The capability description is a resource served by any JPSearch-server. It specifies the capabilities of the repository as well as its custom properties and settings. Any system query option can be enabled or disabled in the capability description. In addition, default values can be overwritten. It is the first resource requested in an initial interaction between a client and a repository. It informs the client about what queries can be send and how they should be formatted for the respective repository.

A JPSearch-client requests resources from a JPSearch-server. A JPSearch-server serves the following resources:

— images: binary image data;

— metadata of images (JPSearch Core metadata or JSON);

— image descriptions (JPOnto);

— collections of images: a set of images (JSON);

— resource identifiers or a collection of resource identifiers: in case the input itself is image data, the repository may return references to related resources of any kind;

— a JPQF output query (XML);

— a capability description: a description of the capabilities and properties of the respective repository (JSON).

All these resources are discussed in more detail further in this part of ISO/IEC 24800.

8 JPSearch API: requesting resources

8.1 Image resources

The most essential resource type of an image repository is an individual image. The return type is a JPEG or JPEG 2000 image file. The original image can be requested by its resource identifier, without specifying any system query options. Various versions of the original image can be requested by

specifying system query options. The following table gives an overview of system query options specific to image resources.

Argument	*Description*	*Values*	*Default*
`crop`	Crops the image to a given aspect ratio. The image is cropped equally to the top and bottom or to the left and right. This option is ignored if a region of interest is specified.	none|w:h where w:h specifies the aspect ratio with w = width and h = height	none
`discardmd`	Specifies whether the embedded metadata should be included or discarded.	Boolean	0
`maxw`	Specifies the maximum width of the returned image.	Unsigned integer	The width of the requested images.
`maxh`	Specifies the maximum height of the returned image.	Unsigned integer	The height of the requested image.
`minw`	Specifies the minimum width of the returned image.	Unsigned integer	The width of the requested images.
`minh`	Specifies the minimum height of the returned image.	Unsigned integer	The height of the requested image.
`quality`	Specifies the quality of the returned image image as a value between 1 and 100 where 1 is the lowest quality and 100 is the highest quality.	Percentage, [1-100]	Original of the requested image, no recompression
`roff`	Region offset. Used in combination with rsize to request a rectangle region of interest.	left,top where top and left are unsigned integers specifying the offset in pixels from the top and the left respectively	0,0
`rsize`	Region size. Used in combination with roff to request a rectangle region of interest.	width,height where width and height are unsigned integers specifying the width and the height of the requested region	The width and height of the requested image.
`scale`	Scale of the returned image with respect to the original image.	Percentage, [0.-100.]	100
`thumb`	Specifies whether the image should be returned as a thumbnail. When a thumbnail is requested, maxw, maxh, minw, minh are set to 256, metadata to discard, crop to 1:1 (square) and quality to 50. These values are overwritten when any of these arguments is specified.	Boolean	0

8.2 Image metadata

Some applications need to present metadata of an image without presenting the image itself. Therefore, metadata can be requested separately from the image by specifying an image URI in combination with the `metadata` system query option. Metadata fields can be selected of the JPSearch Core Metadata schema,

as defined in ISO/IEC 24800-2:2011. Alternatively, additional requestable fields can be defined in the capability description. When the GPSPositioning is requested, it is returned in the geolocation format specified in this document. The value of the metadata argument is a comma-separated list of the requested metadata fields represented by their name or XPath expression. For example, the following request:

```
http://www.repository.org/image.jpg?metadata=Title,Creators,GPSPositioning/@
latitute,GPSPositioning, RightsDescription/Description
```

will return the following fields:

— Title: content of the title field;

— Creators: the GivenName and FamilyName of the creators, space separated;

— GPSPositioning/@latitude: latitude attribute value of the GPS localization;

— GPSPositioning: complete GPS positioning;

— RightsDescription/Description: content of the Description field of the RightsDescription element.

The return format is a JSON file that contains the “metadata” field at root level. The metadata element contains an object with all the requested fields and their values. For example:

```
{
   "metadata":  {
         "Title": "Title of the image",
         "Creators": "John Smith",
         "GPSPositioning/@latitude": "50.85",
         "GPSPositioning": "50.85,4.35",
         "RightsDescripti on/Description": "Rights description"
      }
}
```

Alternatively, when the value is set to all, the complete JPCore metadata is returned. In this case, the return format is XML, i.e. JPCore metadata is returned according to ISO/IEC 24800-2.

Finally, when it is set to `description`, a JPOnto description of the image is returned. In this case, the return format is compliant to the ISO/IEC 24800-2:2011/Amd 1 JPOnto specification.

8.3 Image collections

8.3.1 General

A collection is a set of images identified with a URI. In general, a collection is a subset of all images served by the repository. The repository is not limited in how it manages its own collections. Typically, the path part of the URI can be used to identify repositories. For example, an image hosting service may provide a collection for every user. The collection of a specific user can be identified as follows.

```
http://www.repository.org/username
```

If users can organize their pictures in sets, a set might be identified as follows.

```
http://www.repository.org/username/setname
```

Alternatively, a repository can opt to identify collections using query options, as long as these do not collide with the system query options. For example,

```
http://www.repository.org?user=username
```

The return type of a collection is a JSON (application/json) file listing the resources of the images in the collection. Additional system query options can be specified, for example, to filter the results in the return set. The collection to which these options apply (i.e. the base URI) is referred to as the “context” of the request.

8.3.2 Collection system query options

The following table gives an overview of system query options specific to image collection resources.

Argument	*Description*	*Values*	*Default*
`imgmeta`	Specifies which metadata of the images should be included.	none|*field1,field2,flield3*	none
`top`	Specifies the number of items included in the response.	Unsigned integer	100
`skip`	Specifies the index of the first returned item in the response (starting at 0).	Unsigned integer	0
`orderby`	Specifies by what metadata field the returned results should be ordered.	relevance|*field*	relevance
`orderdirec-tion`	Specifies whether the returned results should be ordered ascending or descending.	asc|desc	desc

8.3.3 Image collection syntax

Image collections are returned as a JSON file with the following top-level fields.

`title`	Title of the collection	String
`meta`	Any custom metadata about the collection as a comma-separated key value-pair list.	String (comma separated key-value pair list)
`count`	The total number of results.	Unsigned integer
`images`	An array of images	Array

Images have the following fields.

`title`	Title of the image	String
`size`	Size of the image	Object with fields width and height which are both unsigned integers representing the width and height of the image respectively.
`uri`	Uri of the image	String (URI)
`meta`	Metadata of the image	String (comma separated key-value pair list)

8.4 Resource identifiers and collections of resources

Some applications need to return other information than images or collections of images. For example, a backend may return a website with information related to a requested image. To allow this kind of interactions and keep compliance with this part of ISO/IEC 24800, any resource can be returned as a resource identifier (URI). In addition, in case multiple resources should be provided, a collection of resources can be returned. However, since this type of resources can be anything and do not fall within the scope of this part of ISO/IEC 24800, no system query options specific to these resources are defined.

Collections of resources adopt the JSON syntax of collections specified in the previous section.

9 API URI-based queries

9.1 General

Querying allows requesting a filtered subset of a repository or a collection by specifying search conditions. Query options can be specified on the repository or on a collection. The search domain is the whole repository or the specified collection, respectively.

The querying system query options provide some arguments for common queries such as location, size or free text. These options are independent form the used metadata scheme. In addition, the `query` system query option allows specifying more complex queries by specifying constraints on metadata fields. However, in this case, the repository should be able to process JPSearch metadata. These keys and the overall metadata model are described in ISO/IEC 24800-2.

9.2 Querying system query options

Argument	*Description*	*Values*	*Default*
`areac`	Return images in the area of the given geolocation.	latitude,longitude with latitude and longitude representing decimal degrees with negative numbers for South and West	not specified
`arear`	Radius of the area of interest. To be used in combination with areac, otherwise this option is ignored.	distance in km	10km
`color`	Return images similar to the given color.	r,g,b where r, g and b represent the red, green and blue channels of the specified color and are values in the interval [0,255]	not specified
`size`	Restricts the size of the returned images to the sizes specified interval value.	[minW:minH-maxW:maxH]	not specified (any)
`text`	Free text search.	String	not specified
`time`	Return only images created since the given time or in the given time interval.	ISO 8601 time or time interval representation	not specified
`type`	Restrict returned images to the given file formats.	jpeg|jpeg2000	not specified
`cond`	See conditional querying.	Query string.	not specified

9.3 Conditional querying

The query system query options allow specifying more complex queries. These queries specify constraints on the values of metadata attributes. These keys and the overall metadata model are described in ISO/IEC 24800-2. Within these queries, several operators and functions can be used.

9.4 Operators

Argument	*Description*	*Example*
`eq`	equals	?cond= JPCore:title **eq** 'sun rise'
`ne`	not equal	?cond=title **ne** null
`gt`	greater than	?cond=JPCore:width **gt** 400
`ge`	greater than or equal	?cond=JPCore:width **ge** 400
`lt`	less than	?cond=JPCore:width **lt** 400

Argument	*Description*	*Example*
`le`	less than or equal	?cond=JPCore:width **le** 400
`and`	and	?cond=JPCore:width gt 400 **and** JPCore:creationDate ge '2013-03-15'
`or`	or	?cond=JPCore:width gt 400 **or** JPCore:creationDate ge '2013-03-15'
`not`	not	?cond=not(value eq null)
`xor`	exclusive or	?cond=JPCore:width gt 400 x**or** JPCore:creationDate ge '2013-03-15'

9.5 String functions

Argument	*Description*	*Example*
`bool contains(s1,s2,cs|1)`	Returns true if s2 is a substring of a1. The optional argument cs indicates whether the function is case sensitive (true by default).	?cond=contains(JPCore:title,'sunrise')
`bool matches(s,re)`	Returns true if s matches with the regular expression re.	?cond=matches(JPCore:title,'.* sunrise')
`string substring(s,start,stop)`	Returns a substring of s starting at position start to position stop. Positions start at index 1 and negative positions start from the back.	?cond=substring(JPCore:title,1,3) eq 'sun'
`string lowercase(s)`	Converts the string s to lowercase.	?cond=substring(JPCore:title,1,3) eq 'sun'

9.6 Date functions

Argument	*Description*	*Example*
`int day(t)`	Returns the day of a timestamp t.	?cond=day(JPCore:creationDate) eq 15
`int hour(t)`	Returns the hour of a timestamp t.	?cond=hour(JPCore:creationDate) eq 5
`int minute(t)`	Returns the minute of a timestamp t.	?cond=minute(JPCore:creationDate) eq 0
`int month(t)`	Returns the month of a timestamp t.	?cond=month(JPCore:creationDate) eq 3
`int second(t)`	Returns the second of a timestamp t.	?cond=second(JPCore:creationDate) eq 30
`int year(t)`	Returns the year of a timestamp t.	?cond=year(JPCore:creationDate) eq 1984
`string weekday(t)`	Returns the weekday of a timestamp t (Monday, Tuesday, Wednesday, Thursday, Firday, Saturday, Sunday)	?cond=weekday(JPCore:creationDate) eq 'Monday'

10 Embedding/Sending data

10.1 Embed types

A request can contain embedded binary data that can be used for querying. In addition, an embedded image can be added to a collection. If metadata has to be uploaded with the image, the metadata should be included within the image in a JPSearch compliant format.

Argument	*Description*	*Values*	*Default*
`embedtype`	Specifies whether binary data is embedded with the request. The data can be included in the post data or as an http file upload (RFC 1867).	none|post|upload	none
`embeddata`	Specifies the type of the embedded data: — image data — standardized descriptor — jpqf query	img|desc|jpqf	img
`embedaction`	Specifies how the embedded data should be used: — return similar images — return related resources — add the (image) data to the collection This option is ignored if `embedtype` is `none` or embeddata is set to `jpqf`.	add|similar| identification| description	similar
`desctype`	The type identification of an embedded descriptor. To be used if `embeddata = desc`, ignored otherwise.	specified in capability description	not specified

10.2 Embedding images

An image or image descriptor can be embedded with the request for several purposes. The supported options are detailed in the following sub-sections.

10.2.1 Add

The `add` option is called in context of a collection where the embedded image is added to the collection. Any metadata associated with the image should be embedded within the image. The response is a status notification. All other system query options are ignored. This option can only be used in combination with embedded image data (`embeddata set` to `img`).

10.2.2 Similar

The `similar` option is used to search for images similar to a given query image. The similar option is called in context of a collection, in other words, the search domain is limited to the collection on which the request is called. The option can be used with an embedded image or an embedded image descriptor. The request returns a collection of images similar to the query image. The return type is a collection of images as specified in this part of ISO/IEC 24800. In case querying system query options are passed, these are executed first, i.e. the search domain is limited to the result set after filtering.

10.2.3 Identification

This `identification` option is used to identify a single main object in an image. It can be used in applications that allow the user to take a photo of an object and identify, for example, to provide related information. The service provider may use the context of a collection to limit the search domain. The option can be used with an embedded image or an embedded image descriptor. The request returns a resource identifier that identifies the main object in the image. Applications that need more advanced identification, such as the identification of multiple objects in a single object, can use the description option.

10.2.4 Description

The `description` option is used to request detailed descriptions of an image. For example, identifying people, buildings or objects in an image. The description option is handled independently of a given collection context and can only be used with an embedded image. The request returns a JPOnto description of the image, compliant to the ISO/IEC 24800-2:2011/Amd 1.

