Infradox Partner API

Technical description and command reference

Document version	8
Date	7 April 2021
By	MW
For	Licensed use only
Status	Final
Location	M:\Infradox\techdoc\Infradox_Partner_API_v4_32.6.odt
API version/build	version 4, build 32.6.0.142 – last build date 9 April 2021
Website	www.infradox.com/infradox-partner-api/

1 Table of contents

2 The Infradox Partner API

2.1 Introduction

The Infradox partner API is a hosted web application that enables third parties to search and retrieve metadata and files for integration in their applications.

The 3rd party API is not intended for backoffice and/or e.g. Wordpress integration. We offer a separate Data access API for that purpose.

Using the partner API is straigthforward. The API accepts both http POST and GET requests and delivers its responses as standardised XML. This document describes how to use the API in detail.

2.2 Support

If you need help, create a ticket on https://xpertise.zendesk.com.
Note that we may not be able to offer support if there are invoices past the due date.

2.3 Terms and conditions

The Infradox partner API is owned and hosted by Xpertise-ICT BV in The Netherlands.

The partner API may not be used without a written license agreement between you, the owner of the database/website that you want to access and Xpertise-ICT BV.

The API may be used to search, to retrieve metadata, to have access to both low and high resolution files etc *in an interactive fashion*. The API may explicitly not be used to copy/download files and/or data for off line use. I.e. the API may not be used in a so called "fire hose" mode where huge amounts of data are downloaded in a single session.

You may not allow anyone outside your company to use the API or its documentation. Any written or non-written information may not be made available to anyone outside your company. The website owner provides the third party with an access key that may be used by a single third party for a single third party application only.

Xpertise-ICT BV has the right to disable your API key if you fail to use the Infradox partner API in accordance with the terms and conditions - or for any other reason.

An API key may be used for a single specific client only. API keys can't be transferred.

Using the API implies that you agree to the terms and conditions.

2.4 Server configuration and adding licences

All Infradox websites are already configured for both internal and 3rd party API use.

To add a licence, find an existing user account in User management (back office) and open the user account properties. Click on the Permissions tab, then click on the "Make API account" button. You can now configure the API account settings and permissions. When you are done click Save and then click on the 'API settings' tab – you'll find the API key at the top. Send this key to your partner.

You can change the settings and permissions at any time but note that changes take effect when your partner creates a new session only. If you want changes to take effect immediately then create a support ticket.

For more detailed information and FAQ's, please visit http://www.infradox.com/infradox-partner-api.

3 Getting started

This chapter provides you with the basic information that you need to use the standardised version of the Infradox partner API.
Note that the URL examples use *https://www.website.com/bin/api.dll*. You will have to replace *www.website.com* with the name of the website on which the API is installed and you may have to change the name of the dll from *api.dll* into another name that will be supplied to you by the website owner.

Generally speaking, your application will use the API to establish a session, execute a search, store the id's that were found and to then retrieve the metadata on a page by page basis.

The API supports both POST and GET requests. The examples in this document use GET requests.

3.1 Establishing a session

Before you can search or use any of the other commands, you'll have to get a session id. You can achieve this by issuing the the *getsessionid* command.

```
https://www.website.com/bin/api.dll/api?cmd=getsessionid
&apikey=EFX1997KKA88&user=0&territory=0
```

- Replace the value for the `apikey=` parameter with the key that you have received. Note that you only include the apikey parameter for the *getsessionid* command.
- * Note that this is the only command that requires the **apikey** parameter. You must not include this parameter for other commands.
- If your account has been configured to require a password, then you must include the parameter pass= followed by your password.
- The response will contain an attribute `<sessionid>`. You will have to store the returned value because it is needed for all other commands.
- If you search using the locations method as described in paragraph 3.3, you can use a single session id to execute requests for different users. In that case it is not necessary nor recommended to create separate session ids for regular use of the API. A single session will stay alive between 2 and 48 hours after the last time it was accessed. The actual expiration time depends on how the server is configured.

Here's an example of the XML response for the *getsessionid* command.

```
<?xml version="1.0" encoding="utf-8"?>
<infradox>
	<response>
		<cmd>getsessionid</cmd>
		<apiversion>32.6.0.142</apiversion>
		<code>70</code>
		<description>created session id</description>
		<remote_name>some website name</remote_name>
		<site_name>websitename</site_name>
		<filtered>0</filtered>
		<sessionid>CB924A34A2794E9DAA097B1C430241-10001998</sessionid>
		<user>0</user>
		<territory>0</territory>
	</response>
</infradox>
```

The code attribute within the response envelope

The XML that the API returns will have the `<response>` envelope as shown above. This envelope has one important attribute; `<code>`. This code will help you interpret the response in case of problems. The *getsessionid* command should return response code 70 which means that a valid session was created on the server. An overview of all possible error and response codes can be found in paragraph 7.1.3.

You can add the sparse=1 parameter to get the <code> attribute only, all other attributes will be removed from the response.

3.2 Testing a session between calls

Before any command and after you have created a session, you can execute the *test* command to ascertain that your session id is still valid.

```
https://www.website.com/bin/api.dll/api?cmd=test
&si=CB924A34A2794E9DAA097B1C430241-10001998&sparse=1
```

The *si*= parameter must specify the session id that you have previously retrieved. Note that all commands require you to specify the si= parameter and a valid session id for its value.

The response code should be 74 to indicate that your session is still valid (see 3.1). Any other code means that you will have to establish a new session before you can execute other commands.

3.3 Searching

The Infradox partner API supports different methods of searching for files. This Getting started chapter describes the preferred and fastest method, that doesn't require you to use a separate session for each end user.

Note that it is also possible to retrieve the metadata immediately with the search command, without first having to retrieve the location id's. This is described in paragraph 3.5.

```
https://www.site.com/bin/api.dll/api?cmd=search
&query=business&parse=locationsxml&locations=1&reverse=1&max=5
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

The search command in the above example retrieves a maximum of 5 locations (max=5) of the last added (reverse=1) files that match the search word "business" (query=business).

Note that locations are physical search engine id's – not file id's. Below is the output for the search command in this example:

```
<?xml version="1.0" encoding="utf-8"?>
<infradox>
	<response>
		<apiversion>32.6.0.142</apiversion>
		<cmd>search</cmd>
		<miliseconds>31</miliseconds>
		<code>101</code>
		<description></description>
		<remote_name>some website name</remote_name>
		<site_name>websitename</site_name>
		<filtered>0</filtered>
		<sessionid>CB924A34A2794E9DAA097B1C430241-10001998</sessionid>
		<user>0</user>
		<territory>0</territory>
	</response>
	<results>
		<data>
			<matchcount>2292</matchcount>
			<matchcounttruncated>5</matchcounttruncated>
```

```
            </data>
            <records>
                  <record>906069</record>
                  <record>906068</record>
                  <record>906067</record>
                  <record>906061</record>
                  <record>906060</record>
            </records>
      </results>
</infradox>
```

A value for the code attribute other then *101*, indicates that there was a problem. You can find an overview of all the response codes in paragraph 7.1.3.

As you can see in the above example, the XML has a `<results>` envelope that contains a `<data>` envelope and a `<records>` envelope. The `<data>` envelope has information with regards to the search results. The `<matchcount>` attribute has the value of records that are in the database for the search query. The `<matchcounttruncated>` is equal to the number of locations that are returned within the XML response.

Important note with regards to location id's

Note that the location id's (these are in fact internal search engine location id's) must not be stored in a database, because these may change. You must use the file id's instead (element #1).

The reverse parameter

You get the locations ordered new-to-old and taken from the top of the database by use of the parameter `reverse=1`. Without this parameter you get the results old-to-new and taken from the bottom of the database. So the `reverse=1` parameter is not just a means of sorting the results in a different fashion, it may return entirely different results if the database holds more than the maximum number of locations that you retrieve.

The csv parameter (recommended)

The `<records>` envelope contains the actual results (location id's). Alternatively, you can retrieve the locations as a single comma separated value by specifying the parameter `csv=1` for the search command. For example:

```
https://www.site.com/bin/api.dll/api?cmd=search
&query=business&parse=locationsxml&locations=1&reverse=1&max=5&csv=1&sparse=1
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

The `<records>` envelope will contain a single record node with a comma separated value that contains all the locations, e.g.:

```
…
<records>
      <record>906069,906068,906067,906061,906060</record>
</records>
…
```

Recommended search parameters

- Limit the number of locations in order to guarantee good search performance. Typical values for the *max=* parameter are in the range of 1,000 to 5,000 locations.
- Use the *csv=1* parameter for a less verbose response by retrieving all the locations as a single comma separated value.

Additional parameters

The API supports a number of additional search parameters for filtering, retrieving latest files etc. This is described in paragraph 5.

Response code 83

If the response code is 83 this means that you have specified parameter values that are not allowed or parameters are missing.

The sparse parameter
The response envelope and the data envelope contain information that may not be important to you. To reduce the size of the response you can include the parameter *sparse=1*

Example without the *sparse=1* parameter

```
<response>
	<apiversion>32.6.0.142</apiversion>
	<cmd>search</cmd>
	<miliseconds>31</miliseconds>
	<code>101</code>
	<description></description>
	<remote_name>some website name</remote_name>
	<site_name>websitename</site_name>
	<filtered>0</filtered>
	<sessionid>CB924A34A2794E9DAA097B1C430241-10001998</sessionid>
	<user>0</user>
	<territory>0</territory>
</response>
<results>
	<data>
		<matchcount>2292</matchcount>
		<matchcounttruncated>3</matchcounttruncated>
	</data>
	<records>
		<record>40157,40156,40149</record>
	</records>
<results>
```

Example with the *sparse=1* parameter (only code, description and sessionid in the response envelope and only matchcount and matchcounttruncated in the data envelope).

```
<response>
	<code>101</code>
</response>
<results>
	<data>
		<matchcount>2292</matchcount>
		<matchcounttruncated>3</matchcounttruncated>
	</data>
	<records>
		<record>40157,40156,40149</record>
	</records>
<results>
```

3.4 Retrieving metadata for a set of location id's

The *getlocationsdata* command is used to retrieve the metadata for a set of location id's that you have retrieved with the search command and the `locations=1` parameter. Your application will have to handle storing the retrieved locations, pagination withn the result set and so on. Applications (web sites) will usually retrieve the metada for a single page at a time. Note that you must store location id's in databases as location id's may not be permanent. Read paragraphs 3.9 and 3.10 for more information about this subject.

```
https://www.site.com/bin/api.dll/api?cmd=getlocationsdata
&parse=locationsdataxml
&elements=1,16,20,90,121
&list=906069,906068,906067,906061,906060
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

- The value for the *list* parameter has the comma separated keys for which you want to retrieve the metadata.

- The value for the *elements* parameter has the comma separated id's of the columns that you want to be included in the XML.
- Empty attributes/values will not be part of the response
- To force strict XML, add the strict=1 parameter to your calls
- To reduce the size of the <response> envelope, add the sparse=1 parameter

The `<results>` envelope will look like this (elements 1,16,20,90 and 121 are included):

```
<records>
  <record location="906060">
    <id>500412</id>
    <caption>Business people in a meeting.</caption>
    <credit>Patrick Somebody</credit>
    <mediadata>mt=0,br=0,nb=0,bi=0,tr=0,mr=1,pr=1,rf=0,rm=1,rs=0,lr=1,ed=0,rc=0,rt=</mediadata>
    <thumbnailpath1>https://www.site.com/cache/tcache/00500412.jpg</thumbnailpath1>
  </record>
  <record location="906061">
    <id>500413</id>
    <caption>Business people in a meeting.</caption>
    <credit>Patrick Somebody</credit>
    <mediadata>mt=0,br=0,nb=0,bi=0,tr=0,mr=1,pr=1,rf=0,rm=1,rs=0,lr=1,ed=0,rc=0,rt=</mediadata>
    <thumbnailpath1>https://www.site.com/cache/tcache/00500413.jpg</thumbnailpath1>
  </record>
  <record location="906067">
    <id>500419</id>
    <caption>Business men working on a laptop.</caption>
    <credit>Patrick Somebody</credit>
    <mediadata>mt=0,br=0,nb=0,bi=0,tr=0,mr=1,pr=1,rf=0,rm=1,rs=0,lr=1,ed=0,rc=0,rt=</mediadata>
    <thumbnailpath1>https://www.site.com/cache/tcache/00500419.jpg</thumbnailpath1>
  </record>
  <record location="906068">
    <id>500420</id>
    <caption>Business men working on a laptop.</caption>
    <credit>Patrick Somebody</credit>
    <mediadata>mt=0,br=0,nb=0,bi=0,tr=0,mr=1,pr=1,rf=0,rm=1,rs=0,lr=1,ed=0,rc=0,rt=</mediadata>
    <thumbnailpath1>https://www.site.com/cache/tcache/00500420.jpg</thumbnailpath1>
  </record>
  <record location="906069">
    <id>500421</id>
    <caption>Business man writing a letter.</caption>
    <credit>Patrick Somebody</credit>
    <mediadata>mt=0,br=0,nb=0,bi=0,tr=0,mr=1,pr=1,rf=0,rm=1,rs=0,lr=1,ed=0,rc=0,rt=</mediadata>
    <thumbnailpath1>https://www.site.com/cache/tcache/00500421.jpg</thumbnailpath1>
  </record>
</records>
```

Elements

You can find an overview of the elements and element groups that you can use in the reference section, paragraph 7.1.

The <mediadata> element is an important element as it indicates restrictions and other important properties. You can find more information in paragraph 7.1.3.

Empty elements
Fields for which there's no value will not be part of the response, regardless of the Elements parameter.

The sparse parameter
You can use the *sparse=1* or *sparse=2* parameter to reduce the size of the response.

Specifying a **value of 1** for the sparse parameter will:

1) remove all attributes except <code> from the reponse envelope.

Specifying a **value of 2** for the sparse parameter will:

1) do what is described above (for sparse=1),

2) replace the attribute names in the record envelope with the letter e followed by the element id. For example <supplierid>123</supplierid> will become <e9>123</e9>. You can find an overview of all element id's in paragraph 7.1.1.

Example when the *sparse=2* parameter is used:

```
<records>
  <record location="906060">
    <e1>500412</e1>
    <e15>Business people in a meeting.</e15>
    <e16>Patrick Somebody</e16>

<e90>mt=0,br=0,nb=0,bi=0,tr=0,mr=0,pr=2,rf=1,rm=0,rs=0,lr=0,ed=1,ce=0,rc=0,rt=0,ta=,tu=,ba=0,bx=0,ns=
0,ps=1,fo=0,or=0,gr=0,nw=0,fex=jpg,fw=2936,fh=1993</e90>
    <e121>https://www.site.com/cache/tcache/00500412.jpg</e121>
  </record>
  …
```

3.5 Searching and retrieving the metadata immediately

As opposed to getting the location id's only as described in paragraph 3.3, you can also execute a search command that will return the metadata in the response immediately.

The url is the same as is described in paragraph 3.3, but

1) you must use the locdataxml template instead of the locationsxml template, ie. replace *parse=locationsxml* with either *parse=locationsdataxml* or *parse=locdataxml,*
2) the *csv* parameter has no function and should be removed from the url,
3) you must add the elements parameter to specify which fields you want to retrieve (for example *elements=1,16,90,120*),
4) the value for the *max* parameter may not be higher than *1000,*
5) you can use the *sparse* parameter with a value of 1 or 2 to reduce the size of the response (as described in the previous paragraph),
6) add strict=1 to force strict XML.

Locdataxml includes the <data /> envelope, for example

```
<data>
       <matchcount>290</matchcount>
       <matchcounttruncated>10</matchcounttruncated>
</data>
```

Locationsdataxml does not include the <data /> envelope.

Example url:

```
https://www.site.com/bin/api.dll/api?cmd=search
&query=flower
&sparse=1
&parse=locdataxml
&locations=1
&reverse=1
&max=3
&elements=90
&si=3ED608EB33124CFFAA3AA75D224825-10001998
```

Response for the above cmd

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<infradox>
	<response>
		<code>101</code>
	</response>
	<results>
		<data>
			<matchcount>161</matchcount>
			<matchcounttruncated>3</matchcounttruncated>
		</data>
		<records>
			<record location="40149">
<e90>mt=0,br=0,nb=0,bi=0,tr=0,mr=0,pr=2,rf=0,rm=1,rs=0,lr=1,ed=1,ce=1,rc=0,rt=0,ta=
,tu=,ba=0,bx=0,ns=0,ps=1,fo=1,or=0,gr=0,nw=0,fex=jpg,fw=1081,fh=1280</e90>
		</record>
			<record location="40156">
<e90>mt=0,br=0,nb=0,bi=0,tr=0,mr=0,pr=2,rf=0,rm=1,rs=0,lr=1,ed=1,ce=1,rc=0,rt=0,ta=
,tu=,ba=0,bx=0,ns=0,ps=1,fo=1,or=0,gr=0,nw=0,fex=jpg,fw=1069,fh=1280</e90>
		</record>
			<record location="40157">
<e90>mt=0,br=0,nb=0,bi=0,tr=0,mr=0,pr=2,rf=0,rm=1,rs=0,lr=1,ed=1,ce=1,rc=0,rt=0,ta=
,tu=,ba=0,bx=0,ns=0,ps=1,fo=1,or=0,gr=0,nw=0,fex=jpg,fw=1030,fh=1280</e90>
		</record>
		</records>
	</results>
</infradox>
```

3.6 Reducing the size of the response

Reducing the size of the response may improve performance as less data has to be sent.

1) Use the sparse parameter as described in the previous paragraphs.
2) Retrieve only the elements that you need.
3) Most websites have static paths to pregenerated web versions of files. If this is the case (request this information for your website by sending e-mail to support@xpertise-ict.com) then do not use retrieve the elements 110,121,122,123 and 124. You can create the paths yourself by retrieving only the 8 digit refcode by adding .jpg and by prepending it with the correct path.

   As an example, instead of retrieving data for elements 121 (thumbnailpath1) and 110 (previewpath) you can simply retrieve element 2 (refcode) and use it like this: https://images.somesite.com/cache/pcache/<refcode>.jpg for the preview file and https://images.somesite.com/cache/tcache/<refcode>.jpg for the thumbnail file.

   You can also find information about the paths in paragraph 7.1.4.

3.7 Strict XML

By default, the generated XML is not strict. I.e. string data may contain characters with accents, ampersands, larger than (>), less than (<) and other characters that are not valid XML. For applications that use a strict XML parser this may cause problems. You can add the strict=1 parameter to your data retrieval commands to force strict XML. Accented characters are replaced with valid entities such as for example ‚ et cetera.

3.8 Retrieving metadata for a preview page with a location id

Since there is no guarantee that a preview image already exists in the cache on the server, you should use the *getmetadata* command when you retrieve data for a preview page. The command retrieves the metadata for the specified file (by its location id) and it will generate a preview file on-the-fly if it doesn't already exist.

```
https://www.site.com/bin/api.dll/api?cmd=getmetadata
```

```
&parse=imgmetadataxml
&elements=
&ir=906069
&locations=1
&sparse=1
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

- The locations=1 parameter tells the API that the value for the ir parameter is a location id.
- You can ommit a value for the elements parameter in combination with the *getmetadata* command and the *imgmetadaxml* template. The resulting XML will then contain all of the available columns. Empty attributes/values are not part of the response.
- Add the sparse=1 parameter to reduce the size of the <response> envelope.
- Add the sparse=2 parameter to reduce the size of the XML document.
- Add the strict=1 parameter to force strict XML values.

3.9 Retrieving metadata for a preview page with a file id

It is important to understand that the commands that are described in the previous paragraphs use location id's. These methods guarantee the fastest possible response times. You must however not permanently store location id's, e.g. when a file is added to an order, a lightbox or a gallery etc. Location id's are not permanent, so you must always use the *id* attribute (element #1) when you want to persist file data.

Fragment of XML response for a search request using location id's:

```
<records>
	<record location="906060"> ← location id – must not be stored in databases
		<id>500412</id> ← file id – can be stored in your database if so required
		<caption>Business people in a meeting.</caption>
		<credit>Patrick Somebody</credit>
		<thumbnailpath1>https://www.site.com/cache/tcache/00500412.jpg</thumbnailpath1>
	</record>
	…
</records>
```

The example below retrieves the metadata for a single file by use of its file id (not its location id) and it makes sure that a preview image exists on the server. Use the ir= parameter to specify the file id.

```
https://www.site.com/bin/api.dll/api?cmd=getmetadata
&parse=imgmetadataxml
&elements=
&ir=906069
&locations=0
&sparse=1
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

- The locations=0 parameter tells the API that the value for the ir parameter is a file id.Ommitting the locations parameter is the same as specifying a value of 0.
- You can ommit a value for the elements parameter in combination with the *getmetadata* command and the *imgmetadaxml* template. The resulting XML will then contain all of the available columns.
- Add the sparse=1 parameter to reduce the size of the <response> envelope.
- Add the sparse=2 parameter to reduce the size of the XML document.
- Add the strict=1 parameter to force strict XML values.

3.10 Retrieving metadata for multiple files using file id's

It is also possible to retrieve metadata for multiple files using a single call using file id's as opposed to location id's (paragraph 3.4). E.g to retrieve the data for files that are stored in an order or a lightbox. The *parse*= parameter must use the metadatalistxml template and the *elements*= parameter must specify the elements you want to retrieve (it cannot be blank). The *list*= parameter must specify the comma separated file id's.

```
https://www.site.com/bin/api.dll/api?cmd=getmetadata
&parse=metadatalistxml
&elements=1,16,20,121
&list=500412,500413,500419,500420,500421
&locations=0
&sparse=1
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

- The locations=0 parameter tells the API that the value for the list parameter are file id's.Ommitting the locations parameter is the same as specifying a value of 0.
- Add the sparse=1 parameter to reduce the size of the <response> envelope.
- Add the sparse=2 parameter to reduce the size of the XML document.
- Add the strict=1 parameter to force strict XML values.

4 Accessing files

4.1 Getting a URL to a comping image

You can use the *getfileurl* command to retrieve the URL (path) to a comping image on the server. The *getfileurl* requires a file id for the *ir=* parameter (not a location id).

```
https://www.site.com/bin/api.dll/api?cmd=getfileurl
&parse=fileurlxml
&ir=00500412
&sizename=800pixels
&um=0
&si=CB924A34A2794E9DAA097B1C430241-10001998
&sparse=1
```

- The getfileurl command requires a value for the sizename= parameter which is 800pixels in the above example. To retrieve a URL to a file without a watermark you must use the correct value for *sizename=* parameter and you must include the *um=1* parameter.
- **Ask the API owner for the available sizes, these are configured on your API user account.**

The XML response will look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<infradox>
	<response>
		<code>101</code>
	</response>
	<results>
		<url>https://www.site.com/851eb935-bfef-4c7a-8338-05d076be33eb/{F8DBF0B7-A656-
4158-A7C4-6770C4E62054}/800pixels/00500412.jpg</url>
	</results>
</infradox>
```

4.2 Downloading high resolution files

The *getfile* command is used to create a copy of a master file in a temporary location on the webserver and to return a URL (path) to that file.

- The API owner needs to have configured your API account with download permissions. If not, the response code will have a value in the 900 range.
- You must check for any restrictions in the metadata before you download a file. That it is possible to download a file does not necessarily mean that it is available to you or your client. The restriction codes are available in the <mediaelement> (#90) attribute. Please refer to paragraph 7.1.3.

```
https://www.site.com/bin/api.dll/api?cmd=getfile
&parse=getfilexml
&ir=00500412
&si=CB924A34A2794E9DAA097B1C430241-10001998
&sparse=1
```

- The *ir* parameter requires an 8 digit value. You can either use element 1 (id) and prepend its value with zeroes to a length of 8 positions, or you can use element 2 (refcode) which is 8 positions always.
- The response will contain the <path> attribute which is the URL that you can use to download the file. The UNC path is not accessible through the Internet. Note that the URL to the high resolution file will be valid for 5 to 20 minutes only (actual time depends on the server configuration).

```
<?xml version="1.0" encoding="utf-8"?>
<infradox>
	<response>
		<code>101</code>
	</response>
	<results>
		<orderid>9506</orderid>
		<filename>WW_00462855.jpg</filename>
		<path>https://www.site.com/851eb935-bfef-4c7a-8338-05d076be33eb/7A923EDF-28BF-
4E49-85DE-2A4A57FF9984/dltnative/WW_00500412.jpg</path>
		<unc>\\IDBXS22\e$\websites\somesite\downloadlink\7A923EDF-28BF-4E49-85DE-
2A4A57FF9984\dltnative\AE_00462855.jpg</unc>
	</results>
</infradox>
```

The *raw=1* parameter

The name of the file to which you'll get a URL depends on how the server is configured for a particular website. A number of actions may take place on the server among them, renaming the file to its original name, prefixing the name with an agency code, IPTC injection and so on. If you want to prevent these steps from being executed you can specify the *raw=1* parameter. In that case the file will be made available with its 8 digit number as a file name.

The *odtguid*= parameter (internal use only)

To prevent an order from being created for a download, you can specify the order detail GUID. This is only available for clients with Infradox websites that use the Infradox partner API to access their own database.

5 Advanced searching

The Infradox partner API supports a number of parameters that you can use for more advanced searches.

5.1 Boolean operators and wildcards

The example shown in 3.3 searches for a single word. The full text search engine however, also supports boolean operators and wildcards.

This document does not describe the search engine in detail but note that you can use AND, OR, AND NOT and you have to use brackets where needed to ascertain that the query is a valid Boolean query.

You can find more information about this subject on the infradox.com website:
https://www.infradox.com/using-the-full-text-search-engine/

You should escape queries so that e.g. spaces are encoded as %20 etc.

Some examples:

```
query=dog or cat
query=(dog or cat) and not wildlife
query=(dog or cat or mouse) and not domestic and not wildlife
```

After you have encoded the query:

```
query=dog%20or%20cat
query=(dog%20or%20cat)%20and%20not%20wildlife
query=(dog%20or%20cat%20or%20mouse)%20and%20not%20domestic%20and%20not%20wildlife
```

You can use a ? for a single position wildcard and a * for a multiple position wildcard. *Note that wildcards searches may be slow on large databases and should be used with caution.Wildcards searches may have been disabled on large databases with huge amounts of indexed words.*

Some examples:

```
query=walk* (may find walk, walker, walked, walks)
query=m?st (may find mast, most, must and mist but it wil not find moist)
query=m*st (may find mast, most, must, mist and moist)
```

Wildcard searches (provided that these are allowed on the server) must match a certain mimimum length setting (depends on the web site). E.g. if the minimum length is 4 then searching for AM* is not allowed but searching for AMS* is allowed.

The default search behaviour is AND. Searching for

dog cat park

is the same as searching for

dog and cat and park

To find pictures of either a dog or a cat in a park the Boolean query would be

(dog or cat) and park

5.2 Standardised search filters

The API supports a number of standardised filters. This was introduced to make it easier to use common search filters regardless of how the filtering system is configured for the website that uses the API. The following filter parameters can be used:

orientations, colors, media, rights, videoprops

The filter values are enumerated values. Multiple values can be supplied for a single parameter by separating the values with commas.

Orientations: 0=Horizontal, 1=Vertical, 2=Square, 3=Panoramic
E.g. **`orientations=0,1`** (only Horizontal and Vertical)

Colors: 0=Color, 1=B&W
E.g. **`colors=1`** (only B&W)

Media: 0=ImagePhoto, 1=Footage, 2=PDF, 3=Document, 4=Audio, 5=Unknown
E.g. **`media=1`** (only Footage)

Rights: 0=RightsManaged, 1=RoyaltyFree, 2=ModelReleased, 3=PropertyReleased, 4=Editorial, 5=NotEditiorial, 5=HighRes, 7=LowRes
E.g. **`rights=0,2,4`** (only Rights Managed with Model Release and marked for Editorial use)

Videoprops: 0=HD, 1=SD, 2=MotionTimeLapse, 3=MotionSlowMotion, 4=MotionRealTime
E.g. **`videoprops=0,4`** (only HD RealtTime video)

You can find more information about this subject on the infradox.com website:
https://www.infradox.com/search-filters-and-access-deny-codes/

5.3 Non-standardised search filters

In addition to the standardised filters that are described in the previous paragraph, many websites support a number of variable filters that can be added to the URL of the search command. Additional filters can be added upon request.

Most of such filter codes are stored in the database as values that start with an @-sign and end with a #-sign. E.g. @SUP123# for supplier 123. You can however not use the @ and # symbols in your query but should use square brackets instead []. You will have to include an additional parameter to tell the API that your request contains one or more filter codes as well. The parameter is **f=1.**

If – for example - you want to search for content with a certain keyword but from supplier 123 only, you can add the supplier's filtercode to the query value. But you'll have to add the f=1 parameter and use brackets instead of the @ and # symbols. So [SUP123] for filter code @SUP123#.

The example below uses a variable filter that is defined on the server with the name _sf1.

```
https://www.site.com/bin/api.dll/api?cmd=search
&query=business&parse=locationsxml&locations=1&reverse=1&max=5
&_sf1=[SUP123]&f=1
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

Contact Xpertise-ICT BV for the availability of search filters on the website for which you are using the Infradox partner API.

5.4 Resetting search filters

Note that certain filters will be persisted in the session object on the server. As a result, such filters are applied to every subsequent search command that uses the same session id. To make sure that filters are cleared it is recommended that you include the filter names (eg _sf1) with a blank value for your search commands.

For example, if you have searched using search filters _sf1, _sf2 and _sf3, then a subsequent search command would include these parameters without values - to make sure the filters are cleared:

```
https://www.site.com/bin/api.dll/api?cmd=search
&query=business&parse=locationsxml&locations=1&reverse=1&max=5
&_sf1=&_sf2=&_sf3=
&si=CB924A34A2794E9DAA097B1C430241-10001998
```

5.5 Latest files

The parameter *latest=1* can be included to retrieve the latest files (last added to the database). The query= parameter will be ignored.

All the files have a code assigned that is generated by the system using the database insertion date and time. You can search for such codes to retrieve files that were added to the system on a certain day, in a certain month etc. These codes always start and end with the letter Q and are known as Q-codes.

For example Q2004111511Q. The format of these codes is YYMMDDHHMM.

You can use a wildcard to ignore certain parts of the codes for instance Q200411*Q means every file that was inserted on 11 April 2020. Q0804*Q means every file that was inserted in April 2020.

You can use these codes in combination with other search queries.
E.g. `(fox or wolf) and Q2101*Q` = fox or wolf – and inserted into the database in January 2021.

5.6 Getting a file count only

You can execute the search command with the *locations=1* parameter as described in paragraph 3.3 to retrieve a file count only by specifying the value 0 for the *max* parameter.

```
https://www.site.com/bin/api.dll/api?cmd=search
&query=business&parse=locationsxml&locations=1&reverse=1&max=0
&si=CB924A34A2794E9DAA097B1C430241-10001998&sparse=1
```

The example below is what the result will look like. Note that the <records> envelope should be ignored. The matchcount value is part of the <data> envelope.

```
<?xml version="1.0" encoding="utf-8"?>
<infradox>
	<response>
		<code>101</code>
	</response>
	<results>
		<data>
			<matchcount>2292</matchcount>
			<matchcounttruncated>0</matchcounttruncated>
		</data>
		<records>
			<record>0</record>
		</records>
	</results>
</infradox>
```

6 Retrieving other data

The Infradox partner API supports retrieval of other data, eg suppliers, orders, invoices and so on. This functionality is not part of the standard usage agreement and further information is available upon request.

7 Reference

7.1 Elements and Element groups

The *getlocationsdata* command uses the elements parameter to specify which data to retrieve. You can specify each element separately but you can also use the predefined group id's.

Build 12.0.0.26 or later supports the *sparse* parameter to reduce the size of the response. The response will have field codes formatted as the letter e followed by the element id as opposed to the element names. For instance <e2>00001234</e2> instead of <refcode>00001234</refcode>. More information can be found in 3.3 and the following paragraphs.

7.1.1 Elements

Id	Name	Description
1	id	The file id
2	refcode	8 digit reference code
3	rowid	Physical rowid ie location
4	fileid	Storage location ID
5	localpath	Internal file name
6	orgfilename	The original file name
8	filters	For internal use
9	supplierid	Id of the supplier
10	copyright	iptc/xmp
11	author	iptc/xmp
12	category	iptc/xmp
13	creation date	iptc/xmp
14	creation time	iptc/xmp
15	caption	iptc/xmp
16	credit	iptc/xmp
17	subcategory	iptc/xmp
18	objectname	iptc/xmp
19	keywords	iptc/xmp
20	headline	iptc/xmp
21	byline	iptc/xmp
22	captionwriter	iptc/xmp
31..50	custom1 .. custom20	20 custom database fields – data will vary depending on the website
60	filewidth0	File properties (media 0)
61	fileheight0	File properties (media 0)
62	filesizemem0	File properties (media 0)
63	filesizedisk0	File properties (media 0)
64	filebpp0	File properties (media 0)
65	filetype0	File properties (media 0)
70	filewidth1	File properties (media 0)
71	fileheight1	File properties (media 0)
72	filesizemem1	File properties (media 0)
73	filesizedisk1	File properties (media 0)
74	filebpp1	File properties (media 0)
75	filetype1	File properties (media 0)
80	datetime	Date and time added to the database
81	datetimetext	Datetime formatted d mmm yyyy hh:nn am/pm
82	date	Datetime formatted yyyymmdd
83	time	Datetime formatted hh:nn
90	mediadata	File restrictions and media type information
91	previewwidth	Width of the preview image in pixels (images only)
92	previewheight	Height of the preview image in pixels (images only)
93	filewidthpixels	Master file width in pixels (images only)
94	fileheightpixels	Master file height in pixels (images only)
95	filewidthinches	Master file width in inches (images only)
96	fileheightinches	Master file height in inches (images only)
97	filewidthcm	Master file width in cm (images only)

Id	Name	Description
98	fileheightcm	Master file height in cm (images only)
99	filesizediskmb	Master file size on disk in MB
110	previewpath	URL to a standard preview file
121	thumbnailpath1	URL to thumbnail size 1
122	thumbnailpath2	URL to thumbnail size 2 (not available for all websites)
123	thumbnailpath3	URL to thumbnail size 3 (not available for all websites)
124	thumbnailpath4	URL to thumbnail size 4 (not available for all websites)
125	Pricecollection (pc)	0 to 9 or -1 if no information is available

7.1.2 Element groups

The predefined element groups are used to retrieve metadata for often used metadata elements. Specifying elements=500 is the same as elements=1,2,3,9,90. Note that you can use more groups and you can combine single elements and element groups. E.g. elements=500,501,

Group id	Remarks	Elements
500	Retrieves the elements that are typically needed for thumbnail pages	1,2,3,9,90
501	IPTC/XMP columns	10,11,12,13,14,15,16,17,18,19,20,21,22
502	All custom fields (1/20)	31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50
503	All properties re online files	110,121,122,6,91,92,93,94,95,96,97,98,99

7.1.3 The mediadata element (90)

The mediadata element has a comma separated value with information re important file properties.

```
<mediadata>
mt=0,br=1,nb=1,bi=1,tr=0,mr=1,pr=0,rf=0,rm=1,rs=1,lr=1,ed=1,rc=0,rt=,ta=18|27|142|
187,tu=,ba=0,bx=0,ns=0,ps=0
</mediadata>
```

property	Description	Information
mt	Media type	0 = images/photos 1 = footage/clips 2 = PDF 3 = Documents other than PDF 4 = Audio 5 = Other media
br	Blocking restriction	1 = The file may not be delivered w/o staff approval
ba	Block agents	1 = File is not available to agents
nb	Non-blocking restriction	1 = Restrictions apply but such restrictions do not prevent a file from being delivered to a client if he/she has such permissions
bi	Block immediate delivery	1 = If an end-user has immediate download permissions (no staff assesment of orders required) – download permissions on an order are blocked until a staff member has explicitly granted permissions
tr	Territorial restrictions	1 = The file is not available in the end user's territory (country or region) Note that this may not be relevant when accessed by a 3rd party. The TA and TU parameters MUST be taken into account, where TA is available in, and TU is unavailable in.
mr	Model Release	1 = Model Release on file, 0=no MR, 2=Not applicable
pr	Property Release	1 = Property Release on file, 0=no PR, 2=Not applicable
rf	Royalty Free	1 = Royalty Free file
rm	Rights Managed	1 = Rights Managed file
rs	Restrictions	1 = The file has restrictions
lr	Low Resolution	1 = Only available as a low resolution file
ed	Editiorial	1 = Editorial use only
ce	Commercial	1 = Commercial use only
ba	Block agents	1 = Not available to agents

bx	Block external API's	1 = May not be accessed by 3rd parties via API's
ns	No Syndication	1 = File may not be syndicated
or	Owner restrictions	Value other than 0; there are owner/supplier restrictions
og	Group restrictions	Value other than 0; there are group imposed restrictions
ps	Print sales	1 = Available for print sales
rc	Restriction count	Indicates the number of additional restrictions besides any other general restrictions which are indicated by br, nb and bi.
nw	NSFW	1 = Not suitable for work
ta	Id's of countries where the file may be sold	Country id's separated with a | This information may or may not be relevant when you access the file with the 3rd party api. If empty, then the file is available worldwide unless there are country id's in the TU property (below)
tu	Id's of countries where the file may not be sold	As above. If tu has a value this means that the file is available everywhere except in the countries specified with this property.
rt	Restriction text	The text of the restriction – eg not available for calendars until 31 December 2015
fex	File extenstion	The extension of the master file filename
fw	Width	Master file width in pixels
fh	Height	Master file height in pixels

It is your responsibility to take restrictions into consideration, i.e. to not download a file if its restrictions indicate that the file may not be available to you or your client. Note that files that are not available within your (or your client's territory) will not be part of any search results provided that the database has been configured to work with territorial restrictions, which is not always the case.

Furthermore third parties will generally use a single session to serve multiple requests without specifying information regarding the end user, in which case territorial restrictions cannot be applied. The `getmetadata` command will still allow you to retrieve data for files that are not available within your territory (or that are otherwise restricted) even if that file is not part of the search results.

7.1.4 Paths to thumbnails

Element id's 121 through 124 are used to retrieve paths to different size thumbnails. The sizes vary between Infradox websites but 121 <thumbnailpath1> is used to display thumbnails for search results etc. Element 122 <thumbnailpath2> is a larger thumbnail (generally 275 pixels on the longest side) that is used for e.g. mouse-overs. Elements 123 <thumbnailpath3> and 124 <thumbnailpath4> are URL's to smaller thumbnails.

The paths to the thumbnails are fixed. If you want the XML output to have a smaller foot print, you can omit elements 121 through 124 and you can then construct the paths to the thumbnails yourself by only retrieving element 2 <refcode> and appending .jpg.

property	Folder	Pixels	Example
<thumbnailpath1>	/cache/tcache	150	https://www.website.com/cache/tcache/00001234.jpg
<thumbnailpath2>	/cache/mcache	275	https://www.website.com/cache/mcache/00001234.jpg
<thumbnailpath3>	/cache/scache	85	https://www.website.com/cache/scache/00001234.jpg
<thumbnailpath4>	/cache/lcache	65	https://www.website.com/cache/lcache/00001234.jpg

Note that the values in the pixels column are an indication only, actual sizes vary between websites.

7.2 Response and error codes

Code	Message
11	Missing cmd
12	Unknown cmd
30	Invalid user id

31	Invalid territory id
32	Invalid API key
70	Created session id
71	Invalid session
72	Session unknown or expired
73	Session w/o valid key
74	Session ok
75	Too many sessions
80	Unknown template
82	File reference error
83	Parameter error (incomplete, missing or otherwise erroneous parameters)
90	Unspecified error - In some cases the actual message for code 90 may be different e.g. the actual exception message
101	OK
105	Received confirmation
106	Failed to process confirmation request
110	Accepting requests
200	No search executed
900	Permissions denied
910	Internal configuration error
920	Download error - In some cases the description for code 920 may be more specific e.g. an exception message like: • *Failed to copy file to download location* • *Not allowed because of restrictions*
922	Maximum number of downloads exceeded
925	Download blocked (the file is not available to you as a result of restrictions)

Example of an error response when trying to download a high resolution / master file:

```
<?xml version=”1.0” encoding=”utf-8”?>
<infradox>
	<response>
		<cmd>getfile</cmd>
		<code>922</code>
		<description>max. number of downloads exceeded</description>
		<remote_name>some third party</remote_name>
		<site_name>infradox</site_name>
		<filtered>1</filtered>
		<sessionid>1BE19A355B444D5CB50FB077032001</sessionid>
		<user>222</user>
		<territory>27</territory>
	</response>
</infradox>
```

Example of an error as a result of an expired or invalid session:

```
<?xml version="1.0" encoding="utf-8"?>
<infradox>
	<response>
		<cmd>getmetadata</cmd>
		<apiversion>32.6.0.142</apiversion>
		<code>72</code>
		<description>session unknown or expired</description>
		<remote_name></remote_name>
		<site_name>arabianeye3</site_name>
		<filtered>0</filtered>
		<sessionid></sessionid>
		<user></user>
```

```
            <territory></territory>
      </response>
</infradox>
```

7.3 The generic XML response envelope

All XML documents that you will retrieve by use of the Infradox partner API have a <response> envelope. It may contain the following attributes:

Name	Description
cmd	The name of the executed command
miliseconds	Execution time in miliseconds
code	A success or error code as described in paragraph 7.1.3
description	A description related to the returned response code. May also contain error messages
remote_name	The name of the remote configuration
site_name	The name of the website for which the API is configured
filtered	Has a value of 1 if the account that is used by the API imposes search filters to exclude certain files from search results
sessionid	The session id that you create with the getsessionid command (3.1) and that you must include in all subsequent requests
user	The id of the the user for which you are executing a command – not required for normal use
territory	The id of the territory of the user for which you are executing a command – not required for normal use

If you add the sparse=1 or sparse=2 parameter to your calls, the response envelope with have the <code> attribute + value only.