Hamamatsu Photonics K.K.

NDP.serve 3

API Reference

3.1.19

Alexander Furnell

July 14, 2017

Contents

1 Introduction **3**

1.1 Making Requests . . . 3
1.2 Parameters . . . 4
1.3 Responses . . . 4
1.4 Errors . . . 5
1.5 Signing in . . . 5

2 NDP.serve Core **7**

2.1 Common Parameters . . . 7
2.2 Functions . . . 8
getversion . . . 8
signin . . . 9
signout . . . 9
whoami . . . 10
getextendeduserinfo . . . 10
getusers . . . 11
getgroups . . . 11
getgroupsandusers . . . 11
getobject . . . 12
getchildren . . . 13
addobject . . . 14
addchild . . . 14
editobject . . . 15
deleteobject . . . 15
deletefiles . . . 16
search . . . 17
grantpermission . . . 18
deletepermissions . . . 18

getpermissions . . . 19
checkpermission . . . 19
hasright . . . 20
2.3 Change Listeners . . . 20
2.3.1 Functions . . . 20
listenforchanges . . . 20
removelistener . . . 21
getchanges . . . 21

3 NDP.serve Image Server 23
3.1 Common Parameters . . . 23
3.2 Functions . . . 24
getversion . . . 24
getimageinfo . . . 25
getimage . . . 26
getregion . . . 27
getmetaimage . . . 28
3.3 File Uploading . . . 29
3.3.1 Functions . . . 29
uploadstart . . . 29
upload . . . 29
uploadstop . . . 30

4 Extra Information 31
4.1 Object Contents . . . 31
4.1.1 Slide . . . 31
4.1.2 Collection . . . 31
4.1.3 File . . . 32
4.1.4 Link . . . 32
4.1.5 Annotation . . . 32
4.1.6 Metadata . . . 32
4.2 Rights . . . 33
4.3 Permissions . . . 33
4.4 Metadata . . . 34

1 Introduction

NDP.serve is a server based software package that allows scanned slide images to be organised and shared in a web-based manner across the Internet/intranets. This document assumes the reader is familiar with the fundamentals of the NDP.serve end-user product.

A user typically interacts with the server using it's built in user interface via a web browser; the API allows other applications to integrate with NDP.serve 3 using HTTP requests.

1.1 Making Requests

Requests are made to the server as follows:

{protocol}://{servername}/{path}/{module}/[api]/]{function}

{protocol} is either 'http' or 'https'

{servername} is the hostname or IP address of the server.

{path} by default is 'ndp' but this is not guaranteed and can be changed by the server administrator.

{module} refers to the module within NDP.serve to communicate with for the core module this would be 'serve' the image server (by default) would be 'imageserver'

Modules that have a GUI (e.g. core and conference) have their API calls under the 'api/' section, those which do not (e.g. the image server) all calls are API calls so they do not use 'api/' in the path.

{function} refers to the specific function to be executed as described later in this document.

1.2 Parameters

NDP.serve is quite flexible with the way parameters to functions are passed to it and will accept any the following for most functions:

- URL encoded parameters in the query string of a GET or POST request.
- URL encoded parameters in the body of a POST request (application/x-www-form-urlencoded).
- Multipart MIME in the body of a POST request (multipart/form-data).
- XML document in the body of a POST request (text/xml).

All text sent to the server must be Unicode encoded in UTF-8 format.

1.3 Responses

If you try to access a non-existent module or function NDP.serve will respond with an HTTP response code of 404 (not found).

If a valid address is specified to an API call NDP.serve will always respond with 200 (OK), it will set the HTTP response header "X-NDP-Response-Status" to 0 for success or 1 for failure. If a response code other than 200 is received something went wrong outside of NDP.serve's control (e.g. network failure, web server error/misconfiguration, NDP.serve service crashed/not running) this should not occur under normal operating conditions.

In the case of failure the body of the response will contain an XML document of type "ndperror" with further details. By not responding with an HTTP error code (e.g. 500) we make sure that the error response reaches the caller and is not intercepted by the web server or any proxy server etc. on route.

In the case of success the response varies based on the function called, most typically will consist of an XML document containing requested data/status of the request or requested image data.

Textual responses will always be in UTF-8 encoded Unicode.

1.4 Errors

As above if NDP.serve could not process the request it will return error XML document, the error document is as follows:

```
<ndperror>
    <code>{integer}</code>
    <text>{string}</text>
    <info>[{string}]</info>
</ndperror>
```

The code specifies a consistent code to identify the error type. The text describes the error type. The info optionally contains further information about the specific error that has occurred.

1.5 Signing in

NDP.serve can be used as a guest to view images that have been made public on the server or signed into for further privileges based on the specific user account.

In either case NDP.serve allocates a "session" that keeps the user signed in or to identify the guest user for remembering settings/keeping track of which guest is which on NDP.conference etc.

Sessions are typically handled as cookies which can generally be handled automatically at the client side by the browser or underlying HTTP implementation however they can also be handled manually in order to gain more control (e.g. if you are connecting from a server based application and need to switch between different users, or a desktop application that allows the user to switch between different accounts that are simultaneously logged on).

The session cookie is named 'ndpSession', it is a session cookies so will last until the user exits the browser (or equivalent). A session lasts for 20 minutes since the last activity or until the user is manually signed-out.

It is optionally possible to have the user automatically sign-in on their next visit to the server. In order to do this a second cookie is created 'ndpLogIn' this is a "one shot" cookie that will sign the user in again (and replace the ndpLogin cookie with a new one). For security if the same ndpLogIn token is used again the user will be signed out of all sessions.

A caller of the API can set the parameter 'sessionid' in any call, this will override the session specified in the cookie (if any).

NDP.serve Core 2

The core has a GUI so API functions can be found under the 'api/' URL, by default the core is at 'serve/' on the server but this could be changed so should be read from the input URL. With everything set to default the URL of an API function is: http[s]://hostname/ndp/serve/api/{function}

2.1 Common Parameters

sessionid – Optional session ID to use for this call (will use value from cookie if not present if not supplied, if no session is specified as either a parameter or cookie, or the session ID is invalid, then a new guest session will be created).

logintoken – Optional login token previously retrieved by using the "remember me" option on a previous "sign in" (will use value from cookie if not present if not supplied). If no valid session is supplied but a valid logintoken is then the user will automatically by signed back in and a new "login token" provided.

myuserid – Optional user ID that the caller expects the call to be executed as, this can be used to prevent fall-back to the guest account if the supplied session ID is invalid (e.g. has expired has been signed out manually). If this parameter is included and the supplied session ID is not valid for the specified user then the call will fail.

2.2 Functions

getversion

Get details of the currently running server

Return Value

```
<versioninfo>
    <service>ndpServeCore</service>
    <name>NDP.serve - Core</name>
    <type>ndpServeCore</type>
    <version>3.0.33</version>
    <major>3</major>
    <minor>0</minor>
    <build>33</build>
</versioninfo>
```

signin

Authenticates a user and returns a session ID to use in future calls, can also be used to create a guest session.

Input Parameters

Name	*Type*	*Description*
username	string	Username of the user to authenticate. If omitted then NDP.serve will attempt to sign the user in from the "login cookie" if present.
password	string	Password for the user to authenticate. Required if the "username" parameter is neither empty nor "guest".
rememberme	bool	If true then upon successful authentication of a user with the supplied username and password combination a login cookie will be set.
setcookie	bool	If true (or not supplied) then the user's new session (and login token if applicable) will be set as cookies.

Return Value

```
<signin>
    <sessionid>New session ID</sessionid>
    <userid>The user ID of the logged user</userid>
    <username>
        The correct username of the signed-in user (may be corrected for
            case vs the one supplied in the request)
    </username>
    <logintoken>
        If 'rememeberme' was specified a login token to allow the user to
            sign back in without re-entering their username/password
    </logintoken>
</signin>
```

signout

Signs the current out the current user by ending the session specified by the sessionid parameter or cookie and invalidates the logintoken (if specified).

Return Value

```
<signout />
```

whoami

Returns the current user based on the standard parameters sent to the function.

Return Value

```
<user>
    <userid>The user's user ID</userid>
    <username>The user's username</username>
</user>
```

getextendeduserinfo

Get extended user information for the current user or the user specified by either the userid, or the sessionid.

Input Parameters

Name	*Type*	*Description*
userid	string	If supplied, the userid to get information for.
sessionid	guid	The user's sessionid that you want information for.
usesession	bool	If this value is true: • For registered users, the userid is taken from the sessionid (if userid is not provided). • For guests, the viewer settings are retrieved from the sessionid.

Return Value

```
<user>
    <userid>User ID</userid>
    <username>Username</username>
    <friendlyname>Name taken from the user's profile: falling back to the
       username</friendlyname>
    <avatar>A number indicating the user's avatar choice</avatar>
    <viewer>{ndp|web}</viewer>
</user>
```

getusers

Returns a list of all the groups and users.

Return Value

```
<users>
    <user>
        <id>User ID</id>
        <username>Username</username>
        <readonly>{1|0}</readonly>
    </user>
    ...
</users>
```

getgroups

Returns a list of all the groups.

Input Parameters

Name	*Type*	*Description*
onlymine	bool	If true, only retrieves the current user's groups.

Return Value

```
<groups>
    <group>
        <groupid>Group ID</groupid>
        <name>Group Name</name>
        <readonly>{1|0}</readonly>
    </group>
    ...
</groups>
```

getgroupsandusers

Returns a list of all the groups and users.

Return Value

```
<groupsandusers>
    <group> ... </group>
    ...
    <user> ... </user>
    ...
</groupsandusers>
```

getobject

Returns an object description from the database.

Input Parameters

Name	*Type*	*Description*
objectid	guid	ID of the object to retrieve, if not supplied then the root object is returned.

Return Value

```
<object>
    <id>Object ID</id>
    <name>Object name</name>
    <type>{slide|collection|file|link|annotation|metadata}</type>
    <searchstring>Text used for search engine</searchstring>
    <dateadded>
        Date the object was added to the database (as Unix timestamp =
            seconds since 1st January 1970)
    </dateadded>
    <permissions>
        [<read/>]
        [<write/>]
        [<saveanno/>]
        [<publishanno/>]
        [<viewrmeta/>]
        [<viewmacro/>]
        [<copyexport/>]
    </permissions>
    <content>
        Contents of the object (specific to the object type).
    </content>
</object>
```

getchildren

Returns a list of the specified object's children.

Input Parameters

Name	*Type*	*Description*
parentid	guid	ID of the object to retrieve the children of, if not supplied then the root object children are returned.
type	string	If specified then limit the results to children that are of this object type.
write	bool	If specified then only return children that the user has write permission on.

Return Value

```
<objects>
    <object> ... </object>
    ...
    <object> ... </object>
</objects>
```

See 'getobject' for a description of the 'object' tags.

addobject

Add's a new object to the database.

Input Parameters

Name	*Type*	*Description*
parentid	guid	ID of the object to add the new object to.
name	string	The name of the object.
type	string	The type of object to add (slide, collection, file, link, etc.).
inherit	bool	Whether to inherit permissions from the parent object or not.
dateadded	timestamp	If provided, set's the object's creation date to this.
owner	string	If provided, set's this object's owner to this ID.
content	xml	The object's content, if any.
searchstring	string	Search syntax to help find this file during searches. If not provided, will use name.

Return Value

```
<object>
    <id>The ID of the added object</id>
</object>
```

addchild

Add's an existing object to the given location.

Input Parameters

Name	*Type*	*Description*
objectid	guid	ID of the object to add to a new location.
parentid	guid	ID of the object to add the existing object to.
inherit	bool	Whether to inherit permissions from the parent object or not.

Return Value

```
<ok></ok>
```

editobject

Edits an existing object in the database.

Input Parameters

Name	*Type*	*Description*
objectid	guid	ID of the object to change.
name	string	If provided, updates the name of the object.
inherit	bool	If provided, updates whether to inherit permissions from the parent object or not.
content	xml	If provided, change the object's content.
searchstring	string	If provided, change the object's search string.

Return Value

```
<object>
    <id>The ID of the edited object</id>
</object>
```

deleteobject

Removes the specified object from the given location. If this is the only place the object is stored, then the object will be removed from the database.

Input Parameters

Name	*Type*	*Description*
objectid	guid	ID of the object to remove.
parentid	guid	ID of the parent to remove the object from.

Return Value

```
<result>
    <status>ok</status>
</result>
```

deletefiles		
Deletes physical files from the image server.		
Input Parameters		
Name	*Type*	*Description*
repositoryid	guid	Repository ID to remove files from.
fileids	string	Comma seperated list of file IDs to delete.
Return Value		
`<success></success>`		

search

Search the database for a list of matching results.

Input Parameters

Name	*Type*	*Description*
in	guid	ID of the object to search within.
for	string	What to search for.
offset	int	If provided, skips this many results.
limit	int	If provided, limits the number of objects returned to this number.
subfolders	bool	Whether to search subfolders. Default is true.
like	bool	Whether to search within partial words. Default is false.
and	bool	Whether to use AND or OR when combining search terms. Default is true.
metadata	string	Which field to search. • "annotations": Search in annotations. • "metadata id": Search in the specific metadata.
filters	string	Space separated list of object types to search.
startdate	int64	Start Date (timestamp).
enddate	int64	End Date (timestamp).

Return Value

```
<objects>
    <object> ... </object>
    ...
    <object> ... </object>
</objects>
```

grantpermission		
Give a user or group a certain permission role on the given object.		
Input Parameters		
Name	*Type*	*Description*
userorgroupid	string	ID of the user or group to give the permission to.
objectid	guid	ID of the object to apply the permission on.
role	int	The role to apply.
Return Value		
<success></success>		

deletepermissions		
Delete all or some permissions from an object.		
Input Parameters		
Name	*Type*	*Description*
objectid	guid	ID of the object to remove permissions from.
userorgroupid	string	If provided, will remove permissions only for this user/-group.
role	int	The specific role to remove, otherwise removes all permissions (if userorgroupid is specified).
Return Value		
<success></success>		

getpermissions

Gets the permission roles associated with the given object.

Input Parameters

Name	*Type*	*Description*
objectid	guid	ID of the object to get permissions for.
explicit	bool	If provided and true, will only get permissions set explicitly on the given object, and not any inherited permissions.

Return Value

```
<objectpermissions>
    <permission>
        <permissionroleid>Role ID</permissionroleid>
        <userorgroupid>User or Group ID</userorgroupid>
    </permission>
    ...
</objectpermissions>
```

checkpermission

Checks if the given ID has the specified permission on an object.

Input Parameters

Name	*Type*	*Description*
objectid	guid	ID of the object to check for permission.
userorgroupid	string	The user or group to check.
permission	int	The permission type to look for.

Return Value

```
<yes></yes>
```

or

```
<no></no>
```

hasright		
Checks if the given ID has the specified rights.		
Input Parameters		
Name	*Type*	*Description*
userorgroupid	string	The user or group to check.
any	bool	Whether any or all rights must be present. Default false.
explicit	bool	Whether a user has the right assigned directly or it can be inherited from the group. Default false.
rights	string	The rights to look for, as a comma separated list.
Return Value		

`<yes></yes>`
or
`<no></no>`		

2.3 Change Listeners

2.3.1 Functions

listenforchanges		
Register a listener for changes (add/edit/remove) of object hierarchies.		
Input Parameters		
Name	*Type*	*Description*
to	guid	The object to listen for changes on.
objecttypes	string	Comma delimited list of object types (e.g. collection).
listentypes	string	Comma delimited list of the types of events to listen for (e.g. add/remove/edit).
timeout	int	How long to listen for changes. Capped at one day for guests. 0 for unlimited.
recursive	bool	Listen to changes inside sub-folders. Default is false.
Return Value		
`<listenerid>Listener ID (guid)</listenerid>`		

removelistener

Remove an object listener. Send with no parameters to remove all of your own listeners.

Input Parameters

Name	*Type*	*Description*
listenerid	guid	If supplied, removes the listener with this ID. Must have Super User right.
userid	string	If supplied, and no listenerid is supplied, removes all listeners for this user. Must have Super User right.

Return Value

```
<ok />
```

getchanges

Gets a list of changes tracked by a listener.

Input Parameters

Name	*Type*	*Description*
to	guid	The ID of the listener.
after	int64	Optional. The "next" attribute from the return XML. Old changes aren't kept cached, so not sending this will not bring up all past changes.

Return Value

```
<changes next="5">
    <change>
        <objectid>{guid}</objectid> (used to be itemid prior to 3.1.37)
        <parentid>{guid}</parentid>
        <objecttype>collection</objecttype>
        <listentype>add</listentype></change>
</changes>
```

NDP.serve Image Server 3

The image server has no GUI so API functions do not use the 'api/' prefix, by default the core is at 'imageserver/' on the server but this could be changed so should be read from the input URL (generally retrieved from a call to 'getobject' on the core). With everything set to default the URL of an image server function is: http[s]://hostname/ndp/imageserver/{function}

3.1 Common Parameters

sessionid – Session ID to use for this call (e.g. as obtained via a 'signin' call to the core).

Image ID

Most calls to the image server will need to refer to a specific image and will thus need to include parameters to indicate which image is required. Normally this will be via an "object ID" but it could also be via a "File ID" and "Repository ID" if the object is not yet added to the database, the user will need to have specific permissions set to be able to access the image server in this way.

objectid – Images will normally be referenced by their object ID obtained from the core.

or

repositoryid – ID of the repository containing the image file.

fileid – The ID of the specific file to be used (obtained from a previous call to "getitems".

3.2 Functions

getversion

Get details of the currently running server

Return Value

```
<versioninfo>
    <service>ndpImageServer</service>
    <name>NDP.serve - Image Server</name>
    <type>ndpImageServer</type>
    <version>3.0.33</version>
    <major>3</major>
    <minor>0</minor>
    <build>33</build>
</versioninfo>
```

getimageinfo

Return various information about an image, generally the first call made when accessing an image.

Input Parameters

Name	*Type*	*Description*
objectid	guid	Alternatively repositoryid & fileid.

Return Value

```
<imageinfo>
    <pixeldimensions>
        <width>Width in pixels</width>
        <height>Height in pixels</height>
    </pixeldimensions>
    <physicalbounds>
        <x>X position of image in nm</x>
        <y>Y position of image in nm </y>
        <width>Width of image in nm</width>
        <height>Height of image in nm </height>
    </physicalbounds>
    <zstack>
        <min>Minimum Z layer in nm</min>
        <max>Maximum Z layer in nm</max>
        <nolayers>Number of Z layers in the image</nolayers>
    </zstack>
    <metadata name="{metadataname}">{metadatavalue}</metadata>
    ...
    <metadata name="{metadataname}">{metadatavalue}</metadata>
    [<focuspoints>
        <point>
            <x>X in nm</x>
            <y>Y in nm</y>
            <z>Z in nm</z>
        </point>
        ...
        <point>
            <x>X in nm</x>
            <y>Y in nm</y>
            <z>Z in nm</z>
        </point>
    </focuspoints>]
</imageinfo>
```

getimage		
Return a scaled overview of the whole image.		
Input Parameters		
Name	*Type*	*Description*
objectid	guid	Alternatively repositoryid & fileid.
width	int	Required - maximum width in pixels of the desired output image.
height	int	Required - maximum height in pixels of the desired output image. *Note: The aspect ratio of the image returned will match that of the source image.*
angle	float or "default"	If supplied rotate the image be the specified angle, if "default" then use the default rotation angle stored in the file (if applicable).
gamma	float or "default"	If supplied apply the specified gamma correction, if "default" then use the default gamma value for the file (if applicable).
brightness	float	If supplied apply the specified brightness value.
contrast	float	If supplied apply the specified contrast value.
red	float	If supplied apply the specified red value.
green	float	If supplied apply the specified green value.
blue	float	If supplied apply the specified blue value.
sharpen	float	If true then apply a sharpen filter to the image.
Return Value		
JPEG stream containing the requested image data.		

getregion		
Return a region of interest from the specified image.		
Input Parameters		
Name	*Type*	*Description*
objectid	guid	Alternatively repositoryid & fileid.
width	int	Required - width in pixels of the output image.
height	int	Required - height in pixels of the output image.
x	int	X coordinate within the source image of the centre of the source ROI in pixels.
y	int	Y coordinate within the source image of the centre of the source ROI in pixels.
sourcewidth	int	Width of the source ROI in pixels.
sourceheight	int	Height of the source ROI in pixels.
angle	float or "default"	If supplied rotate the image be the specified angle, if "default" then use the default rotation angle stored in the file (if applicable).
gamma	float or "default"	If supplied apply the specified gamma correction, if "default" then use the default gamma value for the file (if applicable).
brightness	float	If supplied apply the specified brightness value.
contrast	float	If supplied apply the specified contrast value.
red	float	If supplied apply the specified red value.
green	float	If supplied apply the specified green value.
blue	float	If supplied apply the specified blue value.
sharpen	float	If true then apply a sharpen filter to the image.
Return Value		
JPEG stream containing the requested image data.		

getmetaimage		
Return a scaled overview of the whole image.		
Input Parameters		
Name	*Type*	*Description*
objectid	guid	Alternatively repositoryid & fileid.
width	int	Maximum desired width in pixels of the desired output image.
height	int	Maximum desired height in pixels of the desired output image. *Note: If width and height are not specified then metaimage is returned at its original size.* *If they are specified then the output image will maintain the aspect ratio of the original.*
type	string	Type of metaimage to return: "SlideImage" or "LabelImage".
angle	float or "default"	If supplied rotate the image be the specified angle, if "default" then use the default rotation angle stored in the file (if applicable).
gamma	float or "default"	If supplied apply the specified gamma correction, if "default" then use the default gamma value for the file (if applicable).
brightness	float	If supplied apply the specified brightness value.
contrast	float	If supplied apply the specified contrast value.
red	float	If supplied apply the specified red value.
green	float	If supplied apply the specified green value.
blue	float	If supplied apply the specified blue value.
sharpen	float	If true then apply a sharpen filter to the image.
Return Value		
JPEG stream containing the requested image data.		

3.3 File Uploading

3.3.1 Functions

uploadstart		
Start a chunked upload to the image server.		
Input Parameters		
Name	*Type*	*Description*
filename	string	The name of the file to upload.
size	int64	The file size of the file to upload (for checking completeness when the upload is stopped).
repositoryid	guid	The repository to upload to.
parentfileid	guid	The fileid of a folder to add the file to.
Return Value		
`<uploadid>Upload ID (guid)</uploadid>`		

upload		
Send a chunk of data to the image server.		
Input Parameters		
Name	*Type*	*Description*
uploadid	guid	The upload id given by uploadstart.
pos	int64	The starting position of the data being sent within the file.
size	unsigned int	The amount of data being sent.
Return Value		
`<uploaded></uploaded>`		
Notes		
The above parameters should be encoded in the query string, with the binary data being the only part sent by POST.		

uploadstop		
Finish sending file data to the server.		
Input Parameters		
Name	*Type*	*Description*
uploadid	guid	The upload id given by uploadstart.
force	bool	Whether to stop the upload even if there were problems (if the upload didn't finish, or if moving the file from it's temporary location failed).
Return Value		
<success></success>		

Extra Information 4

4.1 Object Contents

The contents of all objects are stored in the database in XML or plain text, binary objects are stored outside of the database and are linked to in the XML.

The format of the objects are as follows:

4.1.1 Slide

The data for a slide is stored as:

```
<image>
    <repositoryid>{repositoryid}</repositoryid>
    <fileid>{fileid}</fileid>
    <serveraddress>http[s]://{hostname}/{path}</serveraddress>
</image>
```

The server address is not stored in the database (and should not be included when calling 'addobject') but it automatically added by NDP.serve when 'getobject' is called.

4.1.2 Collection

Optional 'collection' element (most standard collections need no content).

If present:

```
<collection>
    [<type>{user|users|group|linked}</type>]
    [<annotations>
        <import>{0|1}</import>
    </annotations>]
    [<repositoryid>
        Repository ID for linked collection
    </repositoryid>]
    [<linkedparent>
        Object ID of parent collection for linked sub-folders
    </linkedparent>]
</collection>
```

The 'annotations' and 'repositoryid' tags only apply to linked collections, the 'linkedparent' tag only applies to linked sub-folders inside linked collections.

4.1.3 File

For uploaded files:

```
<file>
    <filepath>{guid}.{extension}</filepath>
    <mime>{mimetype}</mime>
</file>
```

For files from repositories:

```
<file>
    <filepath>.{extension}</filepath>
    <mime>{mimetype}</mime>
    <repositoryid>{repositoryid}</repositoryid>
    <fileid>{fileid}</fileid>
</file>
```

4.1.4 Link

Plan text URL.

4.1.5 Annotation

'ndpviewstate' XML tag, see 'NDP.view annotation File Format.pdf'.

4.1.6 Metadata

Plain text metadata value.

4.2 Rights

Below are the number values of specific user/group rights:

0 Profile Editing
1 Changing Password
2 Managing Users
3 Changing Server Settings
4 Super User
5 Hosting Conferences

4.3 Permissions

Below are the number values of specific user/group permissions. These generally form part of a role (defined in the NDP.serve administration area), and these roles are assigned to objects.

0 Read
1 Write
2 Saving Annotations
3 Publishing Annotations
4 Reading Protected Metadata
5 Viewing Macro Images
6 Copying and Exporting

4.4 Metadata

Below are the number values of specific built in metadata.

- **-1** Reference
- **-2** Lens
- **-3** Creator
- **-4** Hardware Make
- **-5** Hardware Model
- **-6** Hardware Serial
- **-7** Firmware Version
- **-8** Creation Date/Time
- **-9** Filter Set Name
- **-10** Exposure Ratio
- **-11** Exposure Time
- **-12** Gain Red
- **-13** Gain Green
- **-14** Gain Blue
- **-15** Wave Length
- **-16** Lamp Age
- **-17** Illumination Mode
- **-18** Jpeg Quality
- **-19** Acquisition Time
- **-20** Data Size
- **-21** Compression Type
- **-22** Number of Sub Images
- **-23** Image Format
- **-24** Pixel Width
- **-25** Pixel Height
- **-26** Physical Width
- **-27** Physical Height
- **-28** Number of Layers