

- On the same computer but in a different, for example virtual, environment e.g. Windows OS on a virtual machine (VMware, Parallels)
- On a networked computer e.g. a web server application such as ASP software

For the application to be able to control Horos and consequently facilitate integration it needs to be able to perform all the routine functions around viewing a study on the Horos Database including, but not exclusive to:

- Checking for the presence of the study on the Horos database
- Opening and display images of a specific study
- Deleting a study

Any software/ computer with the network address of your Horos workstation can execute these methods, which can be built-in to Horos or added on as plugins.

To activate the XML-RPC server, which enables Horos to integrate with other information systems go to the *Horos* contextual menu and select *Preferences* (Fig 9.31a). This will bring up the *Preferences* menu window, here you select the *Listener* icon (Fig 9.31b).

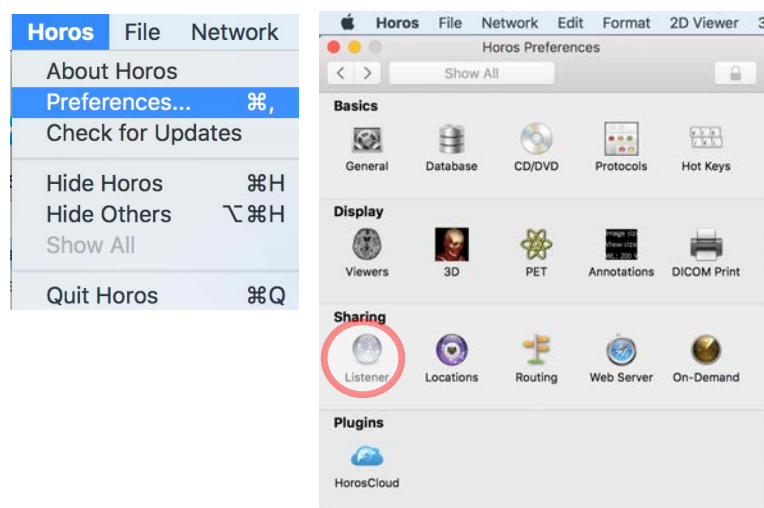


Fig 9.31 The Horos contextual menu with Preferences highlighted (a), and the Listener option in the Preferences window (b).

The XML-RPC server listens to XML\_RPC messages using the 8080 default port. You can change this port to suit your requirements. If you change this port, you must remember to use a port not already in use by another service.

In the Horos Preferences: Listener window select: 'Activate the URL support and HTTP XML-RPC Server on port: xxxx' (Fig 9.32).

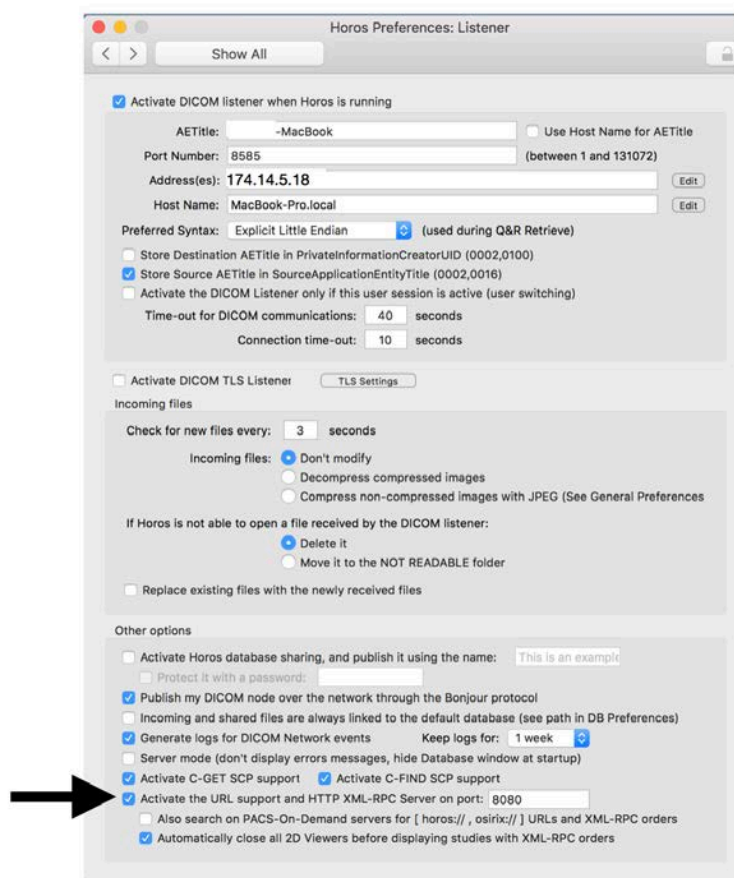


Figure 9.32 *Listener Preferences* with 'Activate URL support and HTTP XML-RPC Server on port: xxxx' highlighted

## Supported XML–RPC Messages in Horos

Horos supports the following XML–RPC messages (built-in):

- The `DBWindowFind` method allows the user to execute an SQL query on the database of Horos and receive the matching elements in the response, see Table 4.1.
- The `CloseAllWindows` method allows the user to close all opened viewers, see Table 4.2.
- The `OpenDB` method allows the user to open a specific database, see Table 4.3.
- The `SelectAlbum` method allows the user to select a specific album and display its content in the database window, see Table 4.4.
- The `GetDisplayed2DViewerSeries` method allows the user to get the UID of the Series displayed in a 2D Viewer, see Table 4.5.

- The `GetDisplayed2DViewerStudies` method allows the user to get the UID of the Studies displayed in a 2D Viewer, see Table 4.6.
- The `Close2DViewerWithSeriesUID` method allows the user to close the 2D Viewer displaying the Series corresponding to the UID, see Table 4.7.
- The `Close2DViewerWithStudyUID` method allows the user to close the 2D Viewer displaying the Study corresponding to the UID, see Table 4.8.
- The `CMove` method allows the user to retrieve a Study from a DICOM server using the Study's accession Number, see Table 4.9.

#### Method Name `DBWindowFind`

Parameters request: a SQL request, see [2] for the syntax. table: an Horos Table, possible values are: Image, Series or Study. execute: Possible values are: Nothing, Select, Open or Delete. Execute is performed at the study level: you cannot delete a single series of a study.

Example request: name == 'Doe' table: Study execute: Select

Example request: name == "(name LIKE '\*DOE\*')" table: Study execute: Open

Response error: 0 elements: an array of elements corresponding to the request

Table 4.1: The `DBWindowFind` method for performing a search in the database.

#### Method Name `CloseAllWindows`

Parameters None

Response error: 0

Table 4.2: The `CloseAllWindows` method for closing all viewers.

#### Method Name `OpenDB`

Parameters path: path of the folder containing the Horos Data folder if path is valid, but if no database is found, Horos will create a new one.

Example path: `/Users/johndoe/Documents/`

Response error: 0

Table 4.3: The `OpenDB` method for opening a specific database.

#### Method Name `SelectAlbum`

Parameters name: name of the album

Example name: Today

Response error: 0

Table 4.4: The `SelectAlbum` method for selecting a specific album.

#### Method Name `GetDisplayed2DViewerSeries`

Parameters None

Response error: 0 elements: array of series corresponding to the displayed windows

Table 4.5: The `GetDisplayed2DViewerSeries` method, for getting the displayed Series.

Method Name `GetDisplayed2DViewerStudies`

Parameters None

Response error: 0 elements: array of series corresponding to the displayed windows

Table 4.6: The `GetDisplayed2DViewerStudies` method for getting the displayed Studies.

Method Name `Close2DViewerWithSeriesUID`

Parameters uid: series instance UID to close

Example uid: 1.3.12.1107.5.1.4.518.4.0.16429682...

Response error: 0

Table 4.7: The `Close2DViewerWithSeriesUID` method for closing a specific Series.

Method Name `Close2DViewerWithStudyUID`

Parameters uid: study instance UID to close

Example uid: 1.2.840.113745.10100.10800.37915.4331...

Response error: 0

Table 4.8: The `Close2DViewerWithStudyUID` method for closing a specific Study.

Method Name `CMove`

Parameters accessionNumber: accessionNumber of the study to retrieve server: server description where the images are located (See Locations preferences)

Example accessionNumber: UA876410 server: Main-PACS

Response error: 0 Possible values for error are: • 0: no error • -1: server not found • -2: server not responding • -3: no study found

Table 4.9: The `CMove` method for retrieving images from a DICOM server.

#### 4.6.3 XML–RPC Messages format

The XML–RPC messages need to be sent in XML format. For example, `Close2DViewerWithStudyUID`, with the parameter uid: 1.2.840.113745.101000.1008000.37915.4331 corresponds in XML to:

```
<?xml version="1.0"?>
<methodCall>
<methodName>Close2DViewerWithStudyUID</methodName>
<params><param><value><struct>
<member>
<name>uid</name>\
<value>
<string>1.2.840.113745.101000.1008000.37915.4331</string>
</value>
</member>
</struct></value></param></params>
</methodCall>
```

For sending non-ASCII characters (like é, ç, ä, . . . ), use UTF–8 encoding:

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

#### XML–RPC Response format

The response to an XML–RPC call is also returned in XML format, for example:

```
<methodResponse><params><param><value><struct>
<member>
<name>error</name>
<value>0</value>
</member>
<member>
<name>elements</name>
<value><array><data><value><struct>
<member>
<name>dateAdded</name>
<value>2010-01-11 11:24:29 +0100</value>
</member>
<member>
<name>scale</name>
<value>0.002500711</value>
</member>
<member>
<name>dateOpened</name>
<value>2010-06-28 11:31:58 +0200</value>
</member>
<member>
<name>seriesSOPClassUID</name>
<value>1.2.840.10008.5.1.4.1.1.128</value>
</member>
<member>
<name>>windowLevel</name>
<value>2444</value>
</member>
<member>
<name>modality</name>
<value>PT</value>
</member>
<member>
<name>name</name>
<value>PET FET Cerebral</value>
</member>
<member>
<name>date</name>
<value>2007-08-03 17:35:47 +0200</value>
</member>
<member>
<name>seriesDICOMUID</name>
<value>1.3.12.2.1107.5.1.4.48545.3</value>
</member>
<member>
<name>id</name>
<value>5</value>
```

```

</member>
<member>
<name>mountedVolume</name>
<value>0</value>
</member>
<member>
<name>numberOfImages</name>
<value>82</value>
</member>
<member>
<name>xOffset</name>
<value>0</value>
</member>
<member>
<name>xFlipped</name>
<value>0</value>
</member>
<member>
<name>displayStyle</name>
<value>3</value>
</member>
<member>
<name>seriesInstanceUID</name>
<value>0005 1.3.12.2.1107.5.1.4.48545.3</value>
</member>
<member>
<name>yFlipped</name>
<value>0</value>
</member>
<member>
<name>yOffset</name>
<value>0</value>
</member>
<member>
<name>windowWidth</name>
<value>3167</value>
</member>
</struct></value></data></array></value>
</member>
</struct></value></param></params></methodResponse>

```

#### Extending the XML-RPC capabilities of Horos

To add specific functions and behaviors to Horos for custom XML-RPC messages, you can write a plugin. Thanks to the XML support in Cocoa (NSXMLDocument), it is very easy to add a method in Horos. An example is available in the Horos Plugins source code repository: XML-RPC-Plugin. In this plugin, two new messages are added to Horos:

##### Method Name Parameters

```

exportSelectedToPath path: /Users/johndoe/Desktop/
openSelectedWithTiling rowsTiling: 2 columnsTiling: 2

```

As shown in the source code of this example, a unique plugin can support multiple messages and parameter decoding is easy:

```
if([[httpServerMessage valueForKey:@"MethodName"]
isEqualToString:@"exportSelectedToPath"])
{
    NSXMLDocument *doc = [httpServerMessage valueForKey:@"NSXMLDocument"];
    NSError *error = nil;
    NSArray *keys = [doc nodesForXPath:@"methodCall/params//member/name"
error:&error];
    NSArray *values = [doc nodesForXPath:@"methodCall/params//member/value"
error:&error];

    if(1 == [keys count] || 1 == [values count])
    {
        int i;
        NSMutableDictionary *paramDict = [NSMutableDictionary dictionary];
        for(i = 0; i < [keys count]; i++)
        {
            [paramDict setValue:[values objectAtIndex:i] objectValue] forKey:[keys objectAtIndex:i] objectValue];
        }

        // Ok, now, we have the parameters -> execute it !

        NSMutableArray *dicomFiles2Export = [NSMutableArray array];
        NSMutableArray *filesToExport;
        filesToExport = [[BrowserController currentBrowser]
filesForDatabaseOutlineSelection:dicomFiles2Export
onlyImages:YES];
        [[BrowserController currentBrowser]
exportDICOMFileInt:[paramDict valueForKey:@"path"]
files:filesToExport
objects:dicomFiles2Export];

        // Done, we can send the response to the sender

        NSString *xml = @"<?xml version=\"1.0\"?>
<methodResponse><params><param><value><struct><member><name>error</name><value>0</value>
</member> </struct></value></param></params></methodResponse>";

        // Simple answer, no errors

        NSError *error = nil;
        NSXMLDocument *doc = [[[NSXMLDocument alloc] initWithXMLString:xml
options:NSXMLNodeOptionsNone
error:&error] autorelease];
    }
}

// To tell to other XML-RPC that we processed this order

[httpServerMessage setValue:doc
forKey:@"NSXMLDocumentResponse"];
```

```
[httpServerMessage setValue:[NSNumber numberWithInt:YES]  
forKey:@"Processed"];
```

## Building a JPEG 2000 Network

This section describes how to build a JPEG 2000 Network, which allows the user to optimize storage and image communication. This can be highly beneficial in radiology to facilitate efficient workflow via fast exchange of DICOM images for reading and interpretation. Horos supports JPEG 2000 syntax but to maintain optimum performance:

- Network bandwidth between the PACS server and workstations should be optimal

A 100MB/s connection should be considered a minimum for efficient image transfer at a rate of 200 CT images/s. Compressed to JPEG 2000 in the JPEG LossLess algorithm, image transfer speed is approximately 2.5 times at faster 500 CT images/s. This is based upon a typical 1000MB ethernet router. To test your network bandwidth go to the Activity Monitor application found in /Applications/Utilities folder.

- Disk speed

Servers need to load and transfer data quickly, for example when multiple workstations request large studies concurrently. Slow disk speeds, for example 5400 RPM, can cause problems in data transfer.

When selecting drives for workstations, a Solid-state drive (SSD) is recommended where possible. It is advisable to use either a RAID 0 or RAID 5 system for image storage on a server. A hardware-based RAID 5 system has the back-up security benefits and for this reason it is recommended for professional use. Eighteen million CT images, equivalent to approximately 9000 studies, each containing 200 images, can be stored on a 4TB system using JPEG2000 LossLess compression.

NAS Systems are not recommended for storing DICOM images due to slow reading and writing of images. The principal benefit of NAS is for multiple workstations to share the content. As only the server needs to load and send images, through the DICOM protocol, this feature is not beneficial.

JPEG2000 LossLess compression aids this process by reducing the data to load by almost three fold. This has the benefit of:

- Greater file storage potential

- Better and faster file reading potential
- Processor speed

JPEG2000 compression/decompression requires processing power. This should be borne in mind when setting up a JPEG2000 PACS system and at least dual core processors on the server side is advisable.

It is the server which compresses the images, before sending the compressed images to Horos workstations. It is the process of adding new images to the server which requires additional processing power, rather than the action of sending them to Horos client workstations.

Horos user workstations will require at least a fast dual-core (3Ghz) or Quad-core system to receive and display images. Current MacBook start at 1.2GHz dual core.

## Setting up and using Horos as a PACS Server

You can set up Horos as a small scale PACS, limiting usage to around 10 workstations. Horos will store and communicate files with other Horos workstations through standard DICOM protocols (C-FIND, C-MOVE and C-GET SCU/SCP). More information on these protocols can be found earlier in this chapter.

### Activating Horos as a server

To activate Horos as a server first activate JPEG2000 compression. To do this go to the *Horos* contextual menu and select *Preferences* (Fig 9.33a). This will bring up the *Preferences* menu window, here you select the *Listener* icon (Fig 9.33b). In the Horos Preferences: Listener window go to the ‘Preferred syntax’ drop down menu and select ‘JPEG2000 Loss less & Lossy’ to activate JPEG2000 compression (Fig 9.34).

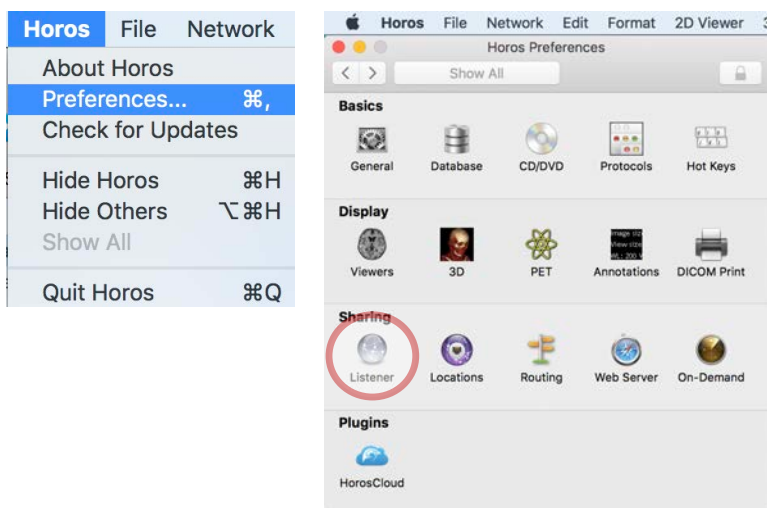


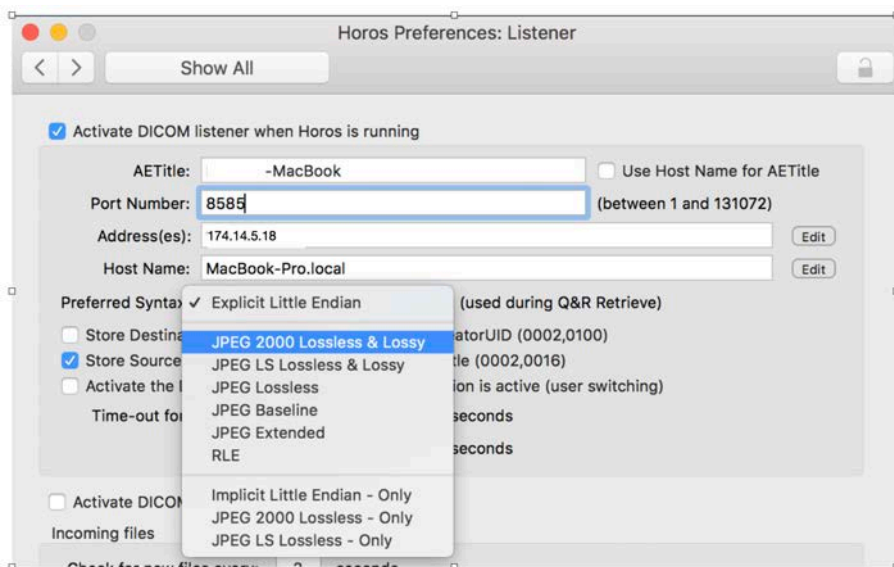
Fig 9.33 The *Horos* contextual menu with *Preferences* highlighted (a), and the *Listener* option in the *Preferences* window (b).

Figure 9.34 Listener Preferences showing the Preferred Syntax drop down menu

Next you need to activate JPEG2000 transfer syntax for each DICOM node. A DICOM node describes any networked DICOM software or hardware, which is used to manage, process or transfer DICOM images. This is essentially a workstation or PACS server. Each DICOM node is uniquely identified by the TCP/IP address of the computer e.g. 174.14.5.18 as well as the TCP/IP Port e.g. 4686 and its Application Entity (AE) title e.g. Horos.

To do this go to the *Horos* contextual menu and select *Preferences* (Fig 9.35a). This will bring up the *Preferences* menu window, here you select the *Locations* icon (Fig 9.35b). In the *Horos Preferences: Location* select JPEG2000 Lossless for each listed DICOM node (Fig 9.36).

