iSite PACS Plug-In Tutorial

iSite PACS 3.6 and 4.1

Rev. 1.02
2008 Oct 28

Philips Healthcare Informatics, Inc.
Enterprise Imaging

PHILIPS

Enterprise Imaging

iSite PACS Plug-In Tutorial

iSite PACS 3.6 and 4.1

Rev. 1.02
2008 Oct 28

Philips Healthcare
is part of
Royal Philips Electronics
www.medical.philips.com
medical@philips.com

North America
Philips Healthcare Informatics
Radiology Informatics Business Unit
4100 East Third Ave., Suite 101
Foster City, CA 94404
USA

Europe
Philips Medical Systems Nederland B.V.
PMS Quality & Regulatory Affairs Europe
Veenpluis 4-6
5684 PC Best
The Netherlands

TMP4.05-02 2008 Apr 28

Table of Contents

Overview

This document contains two tutorials for creating plug-ins for iSite Enterprise and iSite Radiology:

- The ActiveX tutorial shows how to add Plug-ins to iSite Enterprise and iSite Radiology and explains the sample plug-ins provided with the SDK. The iSite Enterprise Client must be installed on your workstation to follow this tutorial.
- The iSite Radiology ActiveX tutorial shows how to add Plug-Ins to iSite Radiology and explains the sample plug-ins provided with the SDK.

Knowledge of HTML and JavaScript is assumed. Refer to the *API Reference Guide* and the sample plug-ins provided with the SDK for more details.

Sample plug-ins are located in the SDK:

```
[SDK INSTALL DIR]\PlugInSamples
```

Related Documents

Please read this and supporting documentation before beginning development with the API. See the following documents for detailed information for all API events and methods:

- *iSite Enterprise ActiveX Reference Guide* for version 4.1/3.6
- *iSite Radiology ActiveX Reference Guide* for version 4.1/3.6

See the following document for details concerning API compatibility between different versions of iSite:

- *iSite PACS API Version Information* for version 4.1/3.6
- *iSite PACS Client and API Overview* for version 4.1/3.6

1 The Plug-In Architecture

Plug-ins are web pages loaded within an Internet Explorer browser control embedded within the Client. They are added and enabled in the iSite PACS Client using the System preference or Machine Preference dialog box. Plug-ins added via the Machine Preferences are workstation specific and will be available to any user that logs on to iSite PACS on that same workstation. Plug-ins added as a System preference will load for all users across an enterprise. The security code feature can be used to limit access to a plug-in.

Note: The **Disable API** check box is selected by default whenever a new plug-in is configured. You must uncheck this if your plug-in will be referencing the iSite PACS API methods and events.

It is important to consider where your plug-in will be used and where the Client will find the components required for the plug-in to function. In general:

- Plug-ins that download all of their content from a web server and are designed to be used by all users in a hospital can be set up as system plug-ins. For example, a global phone directory hosted on a web server within the hospital or a public search engine would be good candidates for System plug-ins.
- Plug-ins that require other applications to be installed on a workstation or rely on other client-side components being installed beforehand should probably be created as Machine based plug-ins. For example, a plug-in that integrates a dictation application into the functioning of iSite PACS would be best suited for a Machine plug-in (since it would not be expected that every workstation accessing iSite PACS would also have the dictation software installed).

The full API is available to a plug-in except for methods used during initialization and login. A web control is created after login, and the iSite Enterprise control is added to the Microsoft Script Engine to provide an interface to the plug-in and the **iSiteEnterprise** ActiveX control. This allows integrators to combine additional functionality to iSite Enterprise (such as adding additional controls in a plug-in web page using JScript). A plug-in web page loads once after a user logs in, and exists until the user logs out.

Preferences for your plug-in can also be stored by adding a Preference page to write configuration information. A Preference page is simply another web page loaded into an Internet Explorer web browser control created within the **Preferences** dialog box. A custom preference page will load initially each time the user opens the **Preferences** dialog box and will unload when the **Preferences** dialog box is closed.

There is no inherent connection or communication between a plug-in and any custom preference page it adds. Utilize the **GetPreference** and **SetPreference** methods and the **EventPreferencesApplied** event in both the plug-in and its preference page to effectively manage the preferences for a plug-in. Preferences for a plug-in should be written as a single XML string and parsed, rather than saving each item as a string. Since the preferences are written to and read from the server, a single XML string per plug-in will provide the best performance. The **DicomFile_ISE.htm** sample shows how to integrate a custom Preference page into a plug-in. See the section "Understanding Events for Preference Dialog Pages" later in this document for more detailed explanation.

Plug-ins use the Microsoft **IDispatchEx** interface, which was specifically designed for JScript. For compatibility with the IDispatchEx interface, plug-ins must contain the script line below even if they contain no script, unless the **Disable API** check box is selected when the plug-in is added.

```
<SCRIPT language="javascript" ></SCRIPT>
```

Plug-ins may need to run ActiveX or scripting, so you must add a "Mark of the Web" comment in the HTML code. This allows the HTML files to be forced into a zone other than the Local Machine zone so that they can then run the script or ActiveX code with a specified security template. This setting works in Internet Explorer 4 and later. To insert a Mark of the Web comment into your HTML file, add the following comment:

```
<!-- saved from url=(0014)about:internet -->
```

Use this comment when you need to generically insert a Mark of the Web. **About:internet** will place the page in the Internet zone.

1.1 Mark of the Web

Plug Ins need to run ActiveX or scripting, so you must add a Mark of the Web comment in the HTML code. This Internet Explorer feature allows the HTML files to be forced into a zone other than the Local Machine zone so that they can then run the script or ActiveX code with a specified security template. This setting works in Internet Explorer 4 and later. To insert a Mark of the Web comment into your HTML file, add one of the following comments:

```
<!-- saved from url=(0014)about:internet -->
```

Use this comment when you generically need to insert a Mark of the Web. About:internet will place the page in the Internet zone.

1.2 Differences Between Plug-ins and an HTML Page

Plug-ins are connected to the Microsoft Script Engine, so the name of the Enterprise control is known, and events are sent to the plug-in. Plug-ins must be coded in JScript, though helper functions written in other scripting languages such as VBScript are supported. You can have many plug-ins, and they can be visible or hidden from the user.

If the plug-in is a dynamic web application and needs to post information back to its web server for server-side processing, use AJAX techniques to accomplish the needed communication.

All events are sent to all plug-ins. Some filtering may need to be done on some events, depending on the requirements of your plug-in. If a plug-in is defined with the Disable API check box, it acts as a normal web page with no access to the iSite PACS API.

You do not need to specify an object block for the enterprise or radiology controls, or initialize or login. The control will already be initialized and a user logged in before the plug-in is executed. The iSite PACS API object names are made available to the script engine of a plug-in if the **Disable API** check box is not selected.

iSite Enterprise example

- iSiteNonVisual.FindExam() (the non visual control)
- iSiteEnterprise.AddViewMenuItem() (the visual control)

iSite Radiology example (only the single control)

- Radiology.FindExam()
- Radiology.AddViewMenuItem()

Event functions are added by creating functions within your web page with “iSiteEnterprise_” or “Radiology_” appended to the front of the event function name.

iSite Enterprise example

```
iSiteEnterprise_EventViewMenuSelected(menuitem, contextRecord)
{
OnViewEvent(menuitem, contextRecord);
}
```

iSite Radiology example

```
Radiology_EventViewMenuSelected(menuitem, contextRecord)
{
OnViewEvent(menuitem, contextRecord);
}
```

Other objects added to the plug-in will require an object block. The DICOM Viewer samples have examples of other objects being used.

1.3 Removing the Sample Plug-ins

1. Bring up the **Preferences** dialog box, and navigate to **Machine Preferences**, **Plug-ins**.
2. Click **Delete** for the plug-ins you wish to delete. The preference page will be automatically deleted.

UnRegister the Plug-in Controls

1. Open up a command prompt window and “cd” or change the directory to:
   ```
   [SDK INSTALL DIR]\PlugInSamples\ISE PlugIn Sample
   ```
2. Run this command to unregister the dll:
   ```
   regsvr32.exe /u WinHelper.dll
   regsvr32.exe /u LogWriter.dll
   ```
3. Change the directory to:
   ```
   [SDK INSTALL DIR]\PlugInSamples\ezDicomMax
   ```
4. Run this command to register the viewer:

```
regsvr32.exe /u ezDICOMax.ocx
```

1.4 Temporarily Disabling all Plug-ins

The iSite PACS API is a powerful tool and if not used correctly, can cause the iSite PACS Client to become unstable or unusable. It is possible for a plug-in, once enabled, to make it impossible to get to the Preferences Dialog to disable the plug-in. For example, a new plug-in causes the iSite PACS Client to crash immediately after login because of a problem in the plug-in's onload functionality. To disable all plug-ins on login, hold down the Ctrl + Alt keys while clicking the Login button on the iSite PACS login page. This will allow you to go to the Preference page and disable/delete any offending plug-in.

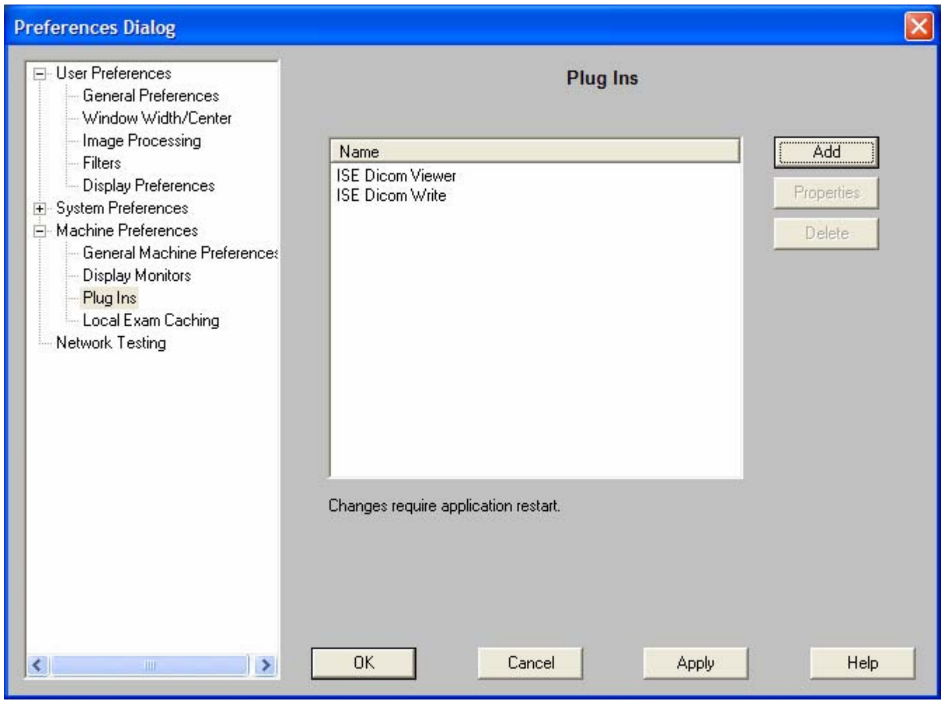
2 iSite PACS ActiveX Plug-in Tutorial

2.1 Configuring the Sample Plug-ins

This tutorial will guide you through enabling two separate plug-ins:

- The first plug-in demonstrates a plug-in that allows a user to write a DICOM file to the local disk based on a custom iSite PACS right-click context menu.
- The second plug-in is a web page containing a public DICOM viewer object that will allow browsing for and displaying the saved DICOM image from the first plug-in.

These are only provided as sample plug-ins to demonstrate some of the iSite PACS plug-in functionality. Unless specified, the steps below are the same for both Enterprise and iSite Radiology.

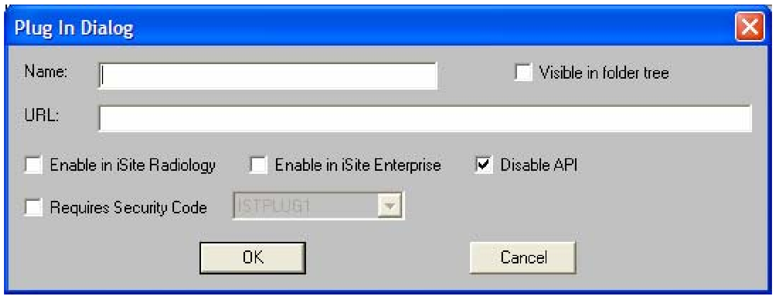


ISE Dicom Viewer
ISE Dicom Write

Changes require application restart.

To add these two sample controls, follow these steps:

1. Bring up the **Preferences** dialog box, and navigate to **Machine Preferences**, then **Plug Ins**.
2. Click **Add**. The **Plug In** dialog box displays.



Visible in folder tree: ☐
Enable in iSite Radiology: ☐
Enable in iSite Enterprise: ☐
Disable API: ☑
Requires Security Code: ☐ ISTPLUG1

3. Depending on the Client in use, name the plug-in either “iSite Enterprise Dicom Write” or “iSite Radiology Dicom Write.” (This value is displayed in the Folder List in the iSite PACS Client.)
4. Select the **Enable in iSite Enterprise** or **Enable in iSite Radiology** check box, depending on which Client you are using to follow the tutorial.
5. Unselect the **Disable API** check box. (This value is checked by default for all new plug-ins. Unselecting this will cause a warning message.)

6. In the URL field, type the location of the sample control web page.

   The default locations for iSite Enterprise would be:

   ```
   [SDK INSTALL DIR]\Plug-inSamples\ISE PlugIn
   Sample\DicomFile_ISE.htm.
   ```

   The default locations for iSite Radiology would be:

   ```
   [SDK INSTALL DIR]\PlugInSamples\ISR PlugIn Sample\DicomFile_ISR.htm.
   ```

7. Click OK.

Note: This is an invisible plug-in, so **Visible in folder tree** is not selected.

Add the second plug-in

1. Click **Add**.
2. Name the plug-in either “iSite Enterprise Dicom Viewer” or “iSite Radiology Dicom Viewer.”
3. Select the **Visible in Folder Tree** check box.
4. Select **Enable in iSite Enterprise** or **Enable in iSite Radiology** check box.
5. In the URL field, type the location where the sample control web page is located.

   The default locations for iSite Enterprise would be:

   ```
   [SDK INSTALL DIR]\ PlugInSamples\ezDicomMax\DicomViewer_ISE.htm.
   ```

   The default locations for iSite Radiology would be:

   ```
   [SDK INSTALL DIR]\ PlugInSamples\ezDicomMax\DicomViewer_ISR.htm.
   ```

6. Click **OK**.
7. Click **OK** on the main **Plug In** dialog box to save these changes.
8. Log out. The plug-in System preferences are read at login and the plug-ins are loaded at login.

2.1.1 Register the Plug-in Controls

The DicomFile plug-ins makes use of two dll files, WinHelper.dll and **LogWriter.dll**. These dll’s and their source code are included to show how to create and use your own custom COM dll’s in iSite PACS plug-ins. The two dll’s also contain functionality that is useful for many different plug-in applications, particularly **LogWriter.dll**. It is recommended that you implement logging functionality in your plug-in for troubleshooting and debugging purposes. Custom COM controls allow a developer to go from the javascript logic in a plug-in to more advanced operations not available via script (for example, writing to an xml file on the local disk, or utilizing another applications API).

1. Open up a command prompt window and “cd” or change the directory to:

   ```
   [SDK INSTALL DIR]\Samples\PlugInSamples\ISE Plug-In Sample
   ```

2. Run these command to register the dll's:

```
regsvr32.exe WinHelper.dll
regsvr32.exe LogWriter.dll
```

3. Change the directory to:

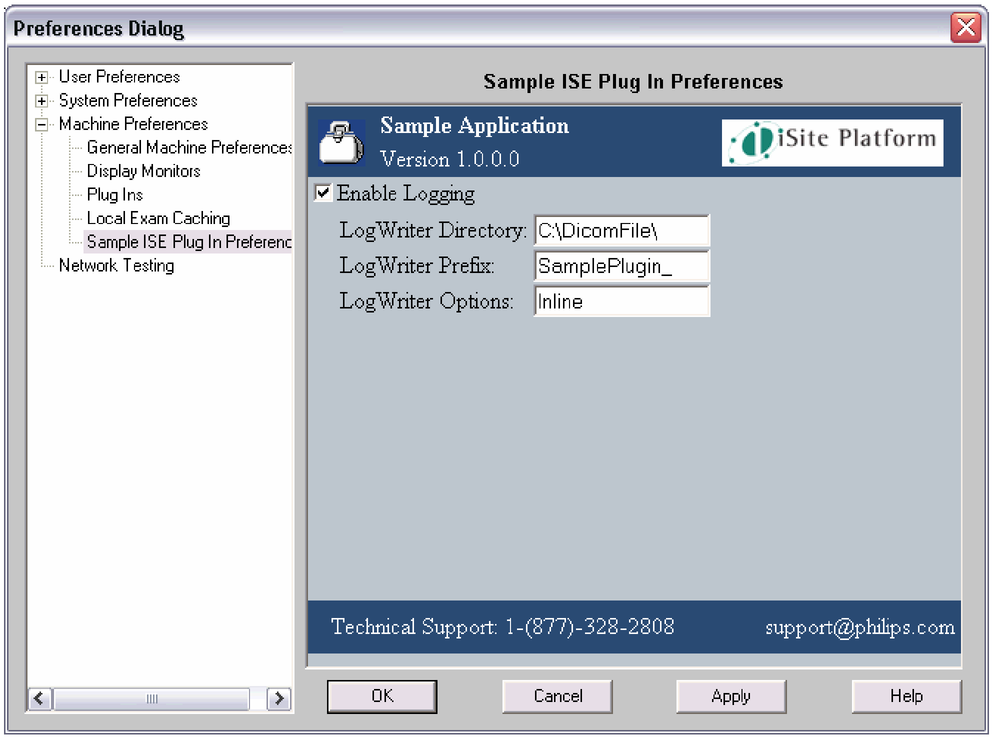
```
[SDK INSTALL DIR]\Samples\PlugInSamples\ezDicomMax
```

4. Run this command to register the viewer:

```
regsvr32.exe ezDICOMax.ocx
```

2.1.2 Login and set the Custom Plug-In Preferences

1. Log in to the Client.
2. Click the **P** icon to open the **Preferences** dialog box.
3. Expand the **Machine Preferences** and click **Sample ISE Plug-in Preference Page** or **Sample ISR Plug-in Preference Page** to view the Custom Plug-in Preference Page.



☑ Enable Logging

LogWriter Directory: C:\DicomFile\

LogWriter Prefix: SamplePlugin_

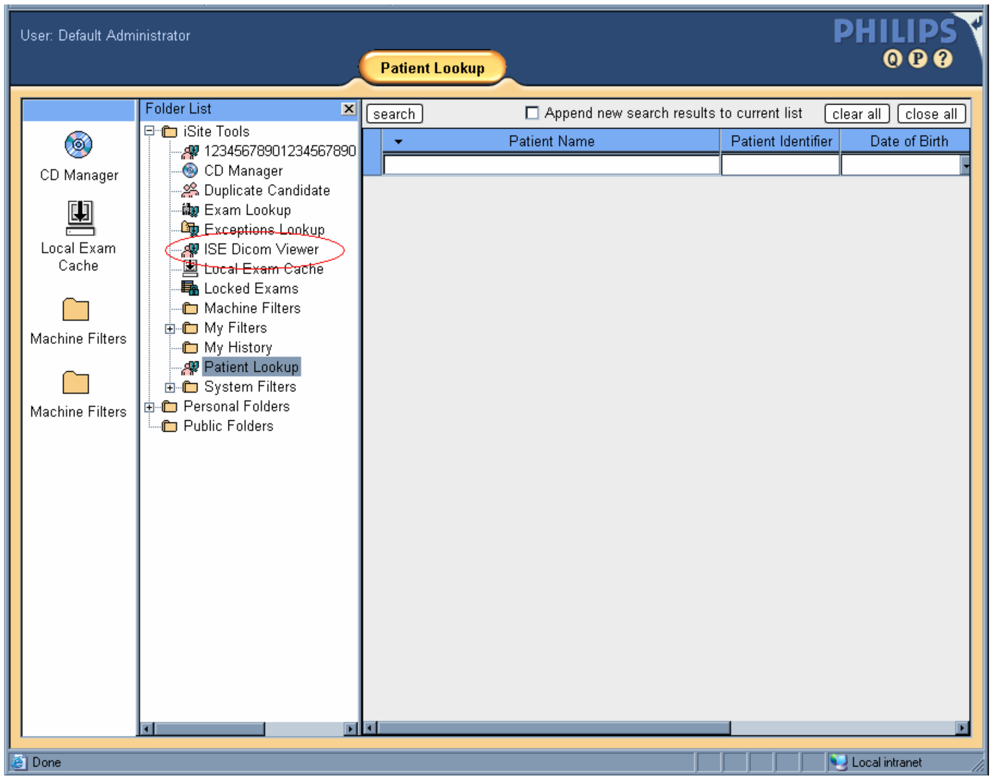
LogWriter Options: Inline

Technical Support: 1-(877)-328-2808 support@philips.com

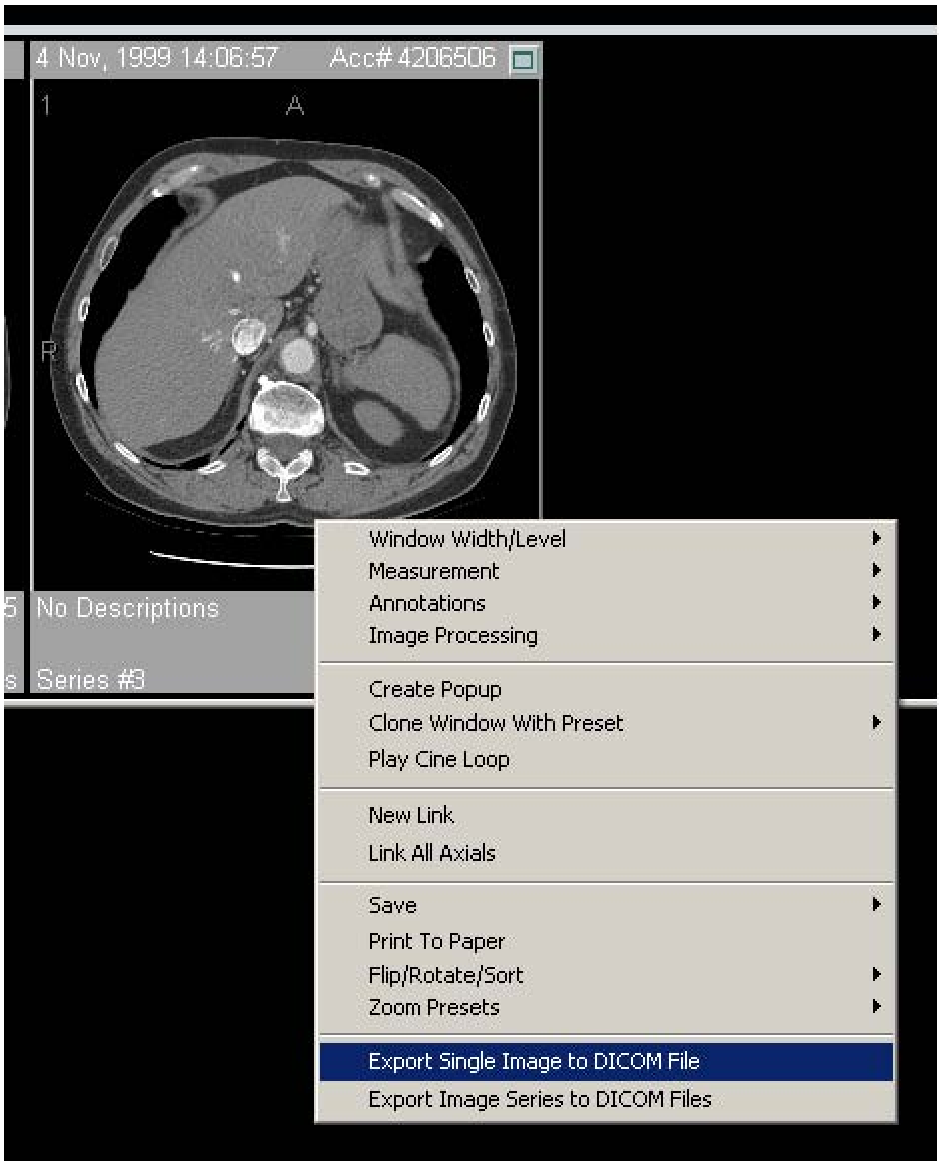
4. To enable plug-in logging, check the checkbox and enter information for the following:
5. **LogWriter Directory:** the directory that the plug-in log will be saved to on your local machine.
   - **LogWriter Prefix**: The prefix prepended to the log file name.
   - **LogWriter Options**: **Inline** makes a log file cleaner by stripping out new line characters.
6. Click **OK** to save the settings.

2.1.3 Testing the Plug-ins

The visible plug-in **iSite Enterprise Dicom Viewer** or **iSite Radiology Dicom Viewer** should be visible in the Folder List:



Navigate to an exam and right click on an image and save the dicom image to disk (any writeable folder will do for this test):



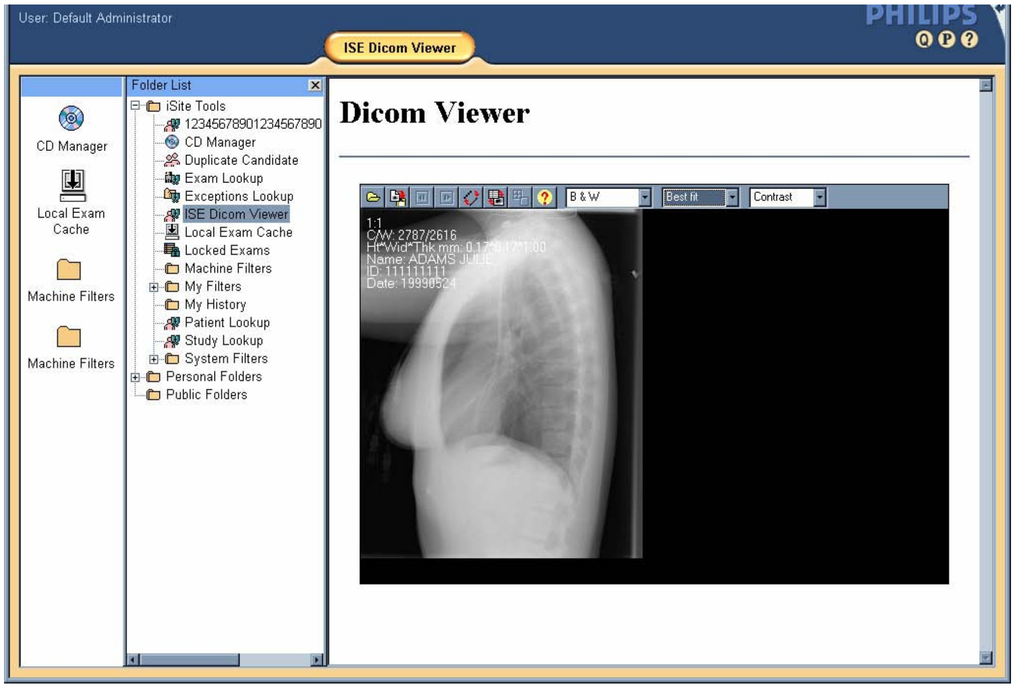
1. The non-visible plug-in was invoked (iSite Enterprise/Radiology Dicom Write) which was passed the DICOM data, and the API call WriteDICOMInstance was used to write the file to disk. View the source code of this plug-in to review how the plug-in works:

   ```
   [SDK INSTALL DIR]\PlugInSamples\ISE PlugIn Sample\DicomFile_ISE.htm
   [SDK INSTALL DIR]\PlugInSamples\ISR PlugIn Sample\DicomFile_ISR.htm
   ```

2. View the DICOM data using the Dicom Viewer Plug-in

   a. Select the “iSite Enterprise/Radiology Dicom Viewer” by selecting it from the folder list

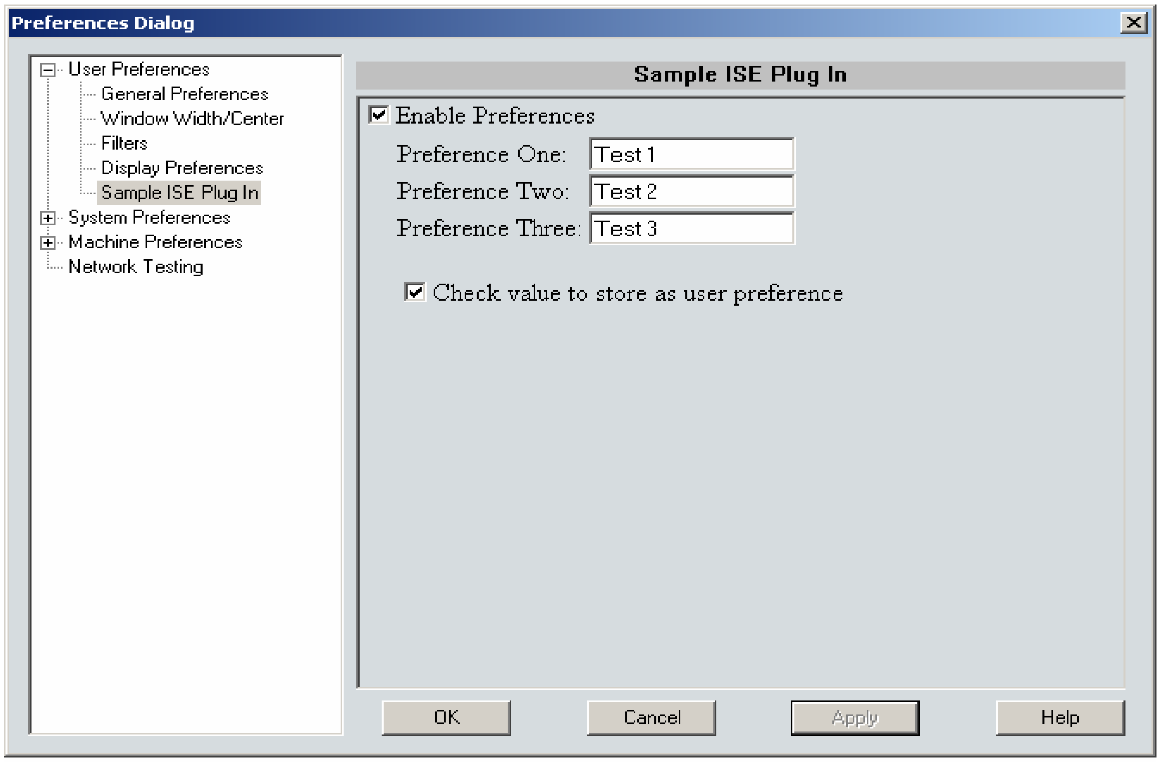
Press the yellow folder icon on the left to point to the directory you saved the image to above and display the DICOM data in the viewer.



2.2 Adding a Preference Page

Although this section only references iSite Enterprise, all information is the same for iSite Radiology as well.

Three sample preference dialog pages were added by the “iSite Enterprise Dicom Write” plug-in: User, System, and Machine. Although the three Preference pages were added, the plug-in only makes use of the Machine Preferences to set the LogWriter preferences. The other pages are added to show where the different pages would reside in the Preferences Dialog. Study this sample to understand how to use a custom preference page with your plug-in. To view the Preference dialog pages, navigate to **Preferences**, **User Preferences**, **Sample ISE Plug-in** to view the sample.



Sample ISE Plug In

☑ Enable Preferences

Preference One:	Test 1
Preference Two:	Test 2
Preference Three:	Test 3

☑ Check value to store as user preference

The source sample for this example is:

```
[SDK INSTALL DIR]\PlugInSamples\ISE PlugIn Sample\ ISESamplePrefsUser.htm.
```

The preference page was added by this code in DicomFile_ISE.htm:

```
function AddPrefPage()
{
// use the prefs file located in this same location
var thispath = document.location.toString();
thispath = thispath.toLowerCase();
var locate = thispath.indexOf("dicomfile_ise.htm");
if (locate > -1)
{
var path = thispath.substring(0, locate);
var prefpath = path + SAMPLE_PREFS_FILE;
iSiteEnterprise.AddPreferencePage(SAMPLE_PREFS_NAME, prefpath,
SAMPLE_PREFS_TYPE);
}
}
```

2.2.1 Understanding Events for Preference Dialog Pages

Special attention should be paid to make sure you are handling events properly in your preference dialog page. To enable the “Apply” button in your dialog, you must detect a change, and enable the Apply button:

```
function SetModified(bFlag)
{
// if the user changes something we need to enable the apply
button
if (m_bModified != bFlag)
{
m_bModified = bFlag;
if (m_bModified == 1)
{
iSiteEnterprise.EnablePreferenceApplyButton();
}
}
}
```

If the user presses the “Ok” or “Apply” button, you will receive an EventPreferencesApply event. This event is where you check the input from the user to make sure it is correct, and you must also filter the event to make sure it is for your preference dialog page, because your plug-in will receive all events, even ones not generated by your preference dialog page. If it is your event, and the input is correct, only then should you write the preferences:

```
function iSiteEnterprise_EventPreferencesApply(PrefPageName,
PrefType)
{
// only do this for our preferences
if ((PrefPageName == SAMPLE_PREFS_NAME) &&
(PrefType == "User"))
{
// no matter what happens we need to clear this flag.
SetModified(0);
// Calls iSiteNonVisual.SetPreference()
SetPreferences();
}
}
```

When the preferences are written to the database, they are updated immediately. If your plug-in must change based on the preferences changing, use the EventPreferencesApplied() in your plug-in to alert you that the preferences have changed.

3 iSite Radiology ActiveX Plug-In Tutorial

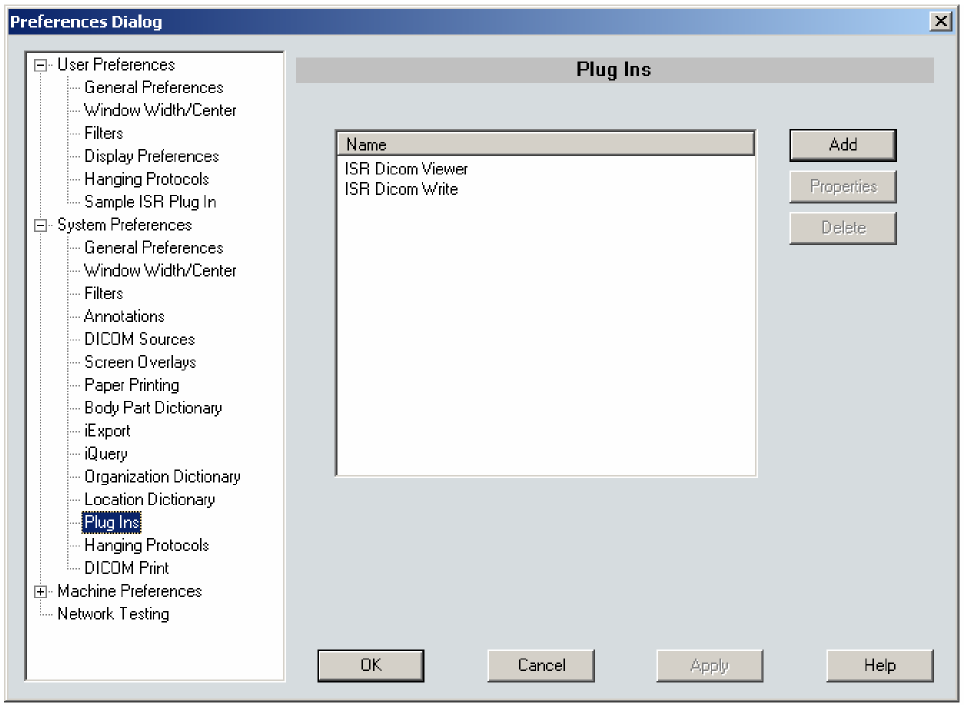
This tutorial shows how to add Plug-Ins to iSite Radiology and explains the sample plug-ins provided with the SDK. Knowledge of HTML and JavaScript is assumed. Please use the API reference guide and the sample plug-ins provided with the SDK for more details.

3.1 Configuring the Sample Plug-ins

Note: If you use the samples as the basis for developing a customized plug-in, be sure to generate new guids for all interfaces and libraries. Use of the guids shown in the sample might cause registry collisions with other plug-ins.

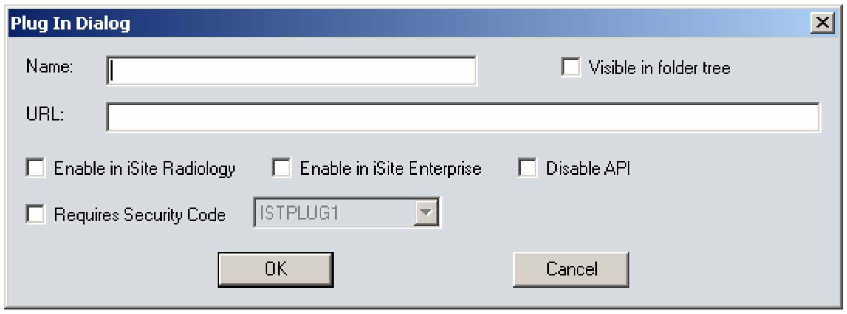
To add these two sample controls, follow these steps:

1. Bring up the **Preferences** dialog box and navigate to **System Preferences**, **Plug-ins**:



Name
ISR Dicom Viewer
ISR Dicom Write

2. Click **Add**. The following dialog box displays.



3. In the Name field enter **ISR Dicom Write**.
4. Select the **Enable in iSite Radiology** check box.
5. Make sure that the **Disable API** check box is unselected. Close the warning message.
6. In the URL field, enter thethe location where the sample control Web page is located. The default location would be:

   ```
   [INSTALL DIR]/Samples/PlugInSamples/ISR PlugIn
   Sample/DicomFile_ISR.htm.
   ```

   This plug-in writes a DICOM file to the local disk when the user brings up the view menu over an image, and selects either “Export Single Image to DICOM File” or "Export Image Series to DICOM Files." This is only provided as a sample. A real plug-in would be located in a globally accessible location (such as a Web server) and not the local disk. This plug-in is dependant on the **ISRSamplePrefs.htm** and **ISRSamplePrefs.js** files in the same directory.
7. Click **OK**.

Note: This is an invisible plug-in, so **Visible in folder tree** is not selected.

Add the second plug-in:

1. Click **Add**.
2. In the **Name** field enter **ISR Dicom Viewer**.
3. Select the **Visible in Folder Tree** check box.
4. Select the **Enable in iSite Radiology** check box.
5. Make sure the **Disable API** check box is selected (it should be by default).
6. In the URL field enter the location where the sample control Web page is located. The default location would be:

   ```
   [INSTALL DIR]/Samples/PlugInSamples/ezDicomMax/DicomViewer_ISR.htm.
   ```

   This plug-in is a public domain DICOM viewer that is provided only as a sample to show you how to add a plug-in, and provides a companion sample to the plug-in added above. (After the DICOM file is saved the DICOM image can be viewed from this plug-in.)

8. Click **OK**.
9. Click **OK** on the main Plug-in dialog box to save these changes.
10. Log out. The Plug-in System preferences are read at login, and the plug-ins are loaded at login.

Register the Plug-in Controls

The **DicomFile_IS.htm** plug-in makes use of two dll files, **WinHelper.dll** and **LogWriter.dll**. These dll's and their source code are included to show how to create and use your own custom COM dll's in iSite PACS plug-ins. The two dll's also contain functionality that is useful for many different plug-in applications, particularly **LogWriter.dll**. It is recommended that you implement logging functionality in your plug-in for troubleshooting and debugging purposes.

1. Open up a command prompt window and "cd" to:

   ```
   [INSTALL DIR]\Samples\PlugInSamples\ISR PlugIn Sample
   ```

2. Run this command to register the dll:

   ```
   regsvr32.exe WinHelper.dll
   regsvr32.exe LogWriter.dll
   ```

3. Navigate to the following directory:

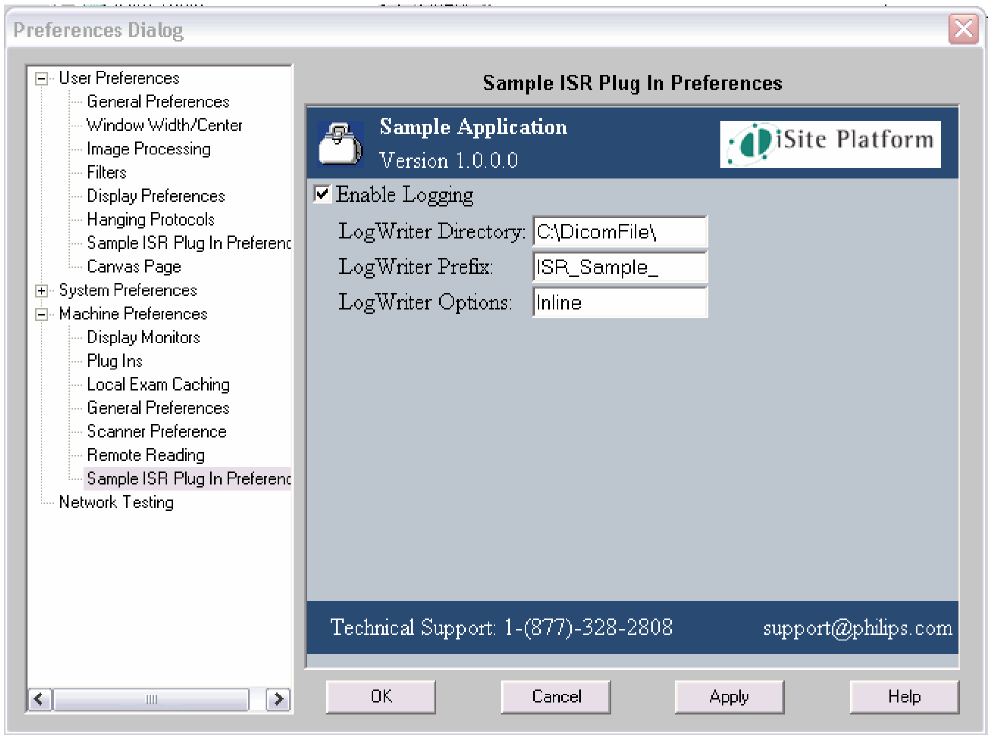
   ```
   [INSTALL DIR]\Samples\PlugInSamples\ezDicomMax
   ```

4. Run this command to register the viewer:

   ```
   regsvr32.exe ezDICOMax.ocx
   ```

Login and set the Custom Plug-In Preferences

1. Login to iSite Radiology.
2. Click the **P** icon to open the **Preferences** dialog box
3. Expand the **Machine Preferences** and click **Sample ISR Plug-in Preference Page** to view the Custom Plug-in Preference Page.



[✓] Enable Logging

LogWriter Directory: C:\DicomFile\

LogWriter Prefix: ISR_Sample_

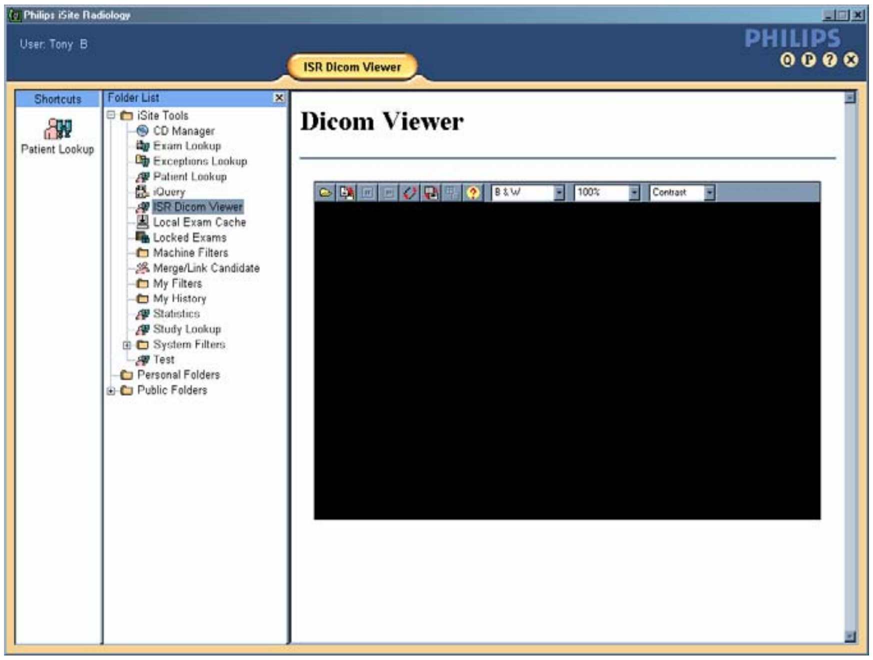
LogWriter Options: Inline

Technical Support: 1-(877)-328-2808 support@philips.com

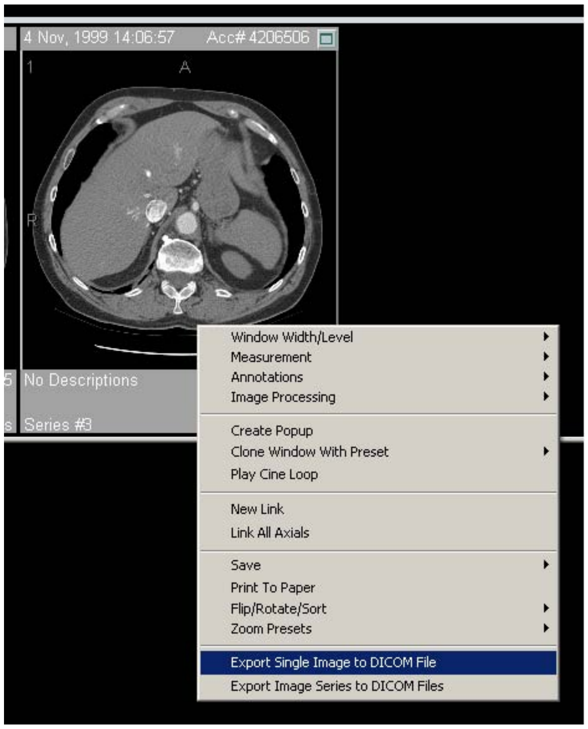
4. To enable plug-in logging, select the check box and enter information for the following:
   - **LogWriter Directory**: The directory that the plug-in log will be saved to on your local machine.
   - **LogWriter Prefix**: The prefix prepended to the log file name.
   - **LogWriter Options**: **Inline** makes a log file cleaner by stripping out new line characters.
5. Click **OK** to save the settings.

Test the Plug-ins:

The visible plug-in **ISR Dicom Viewer** should be visible in the Folder List:



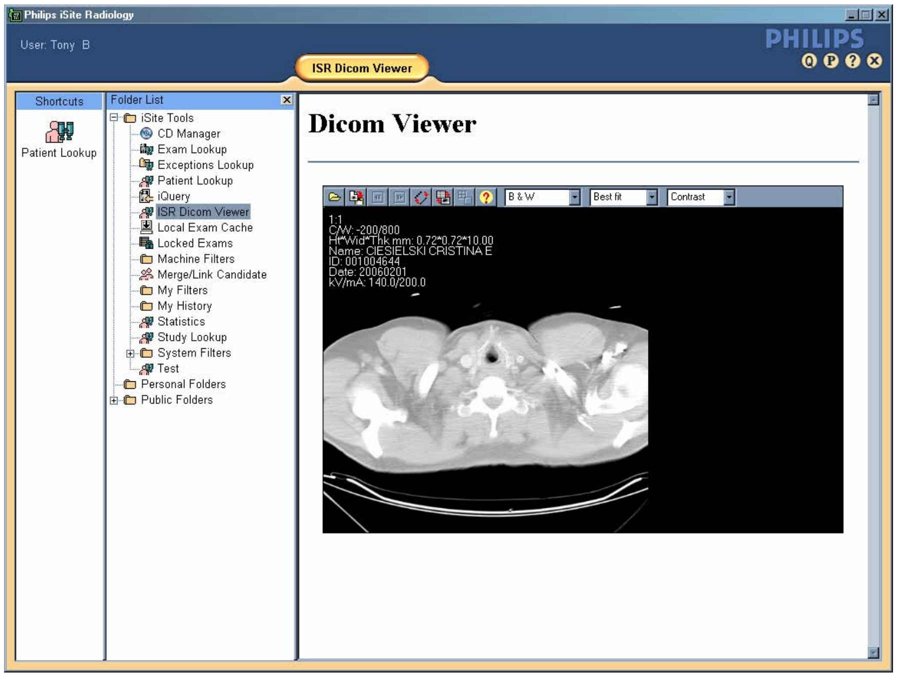
1. Navigate to an exam and right click on an image and save the dicom image to disk (any write able folder will do for this test):



The non-visible plug-in was invoked (ISR Dicom Write) which was passed the DICOM data, and the API call **WriteDICOMInstance** was used to write the file to disk. View the source code of this plug-in to review how the plug-in works:

```
[INSTALL DIR]\Samples\PlugInSamples\ISR PlugIn
Sample\DicomFile_ISR.htm
```

2. View the DICOM data using the ISR Dicom Viewer Plug-in.
3. Select the **ISR Dicom Viewer** by selecting it from the Folder List,
4. Press the yellow folder icon on the left to point to the directory you saved the image to above and display the DICOM data in the viewer.



```
1:1
C/W: -200/800
Ht*Wid*Thk mm: 0.72*0.72*10.00
Name: CIESIELSKI CRISTINA E
ID: 001004644
Date: 20060201
kV/mA: 140.0/200.0
```

The visible plug-in was invoked (ISR Dicom Viewer). View the source code of this plug-in to review how the plug-in works:

```
[INSTALL DIR]\Samples\PlugInSamples\ezDicomMax\ DicomViewer_ISR.htm
```

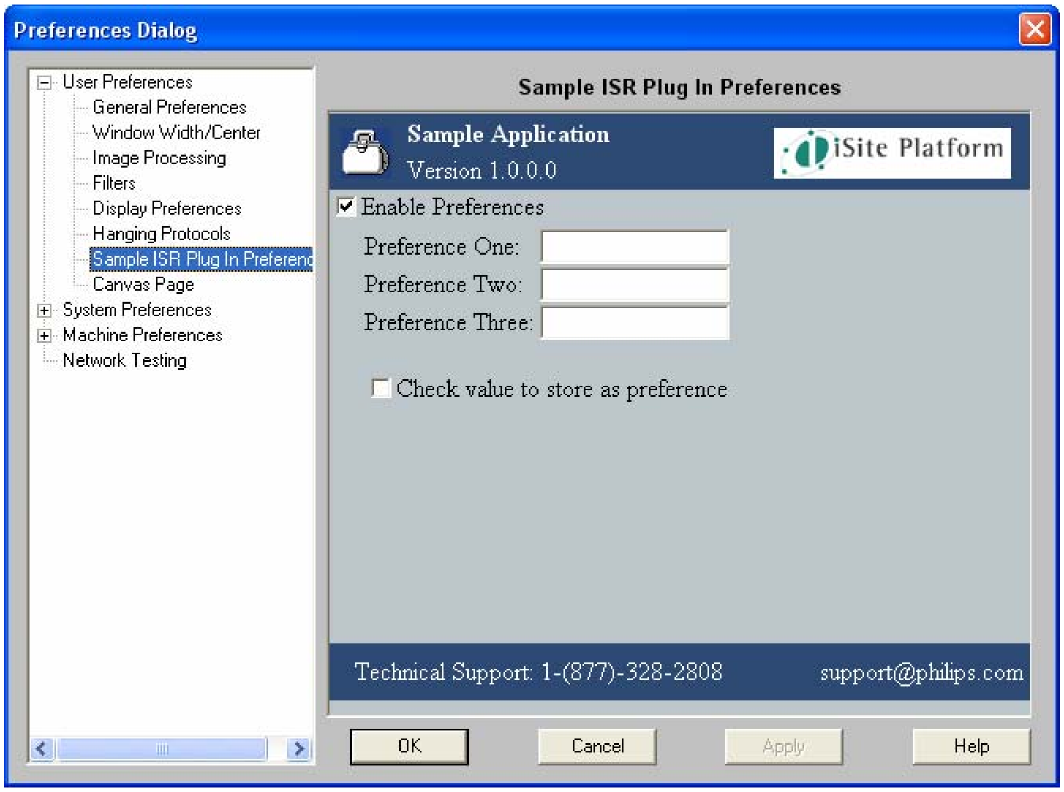
Note: This plug-in uses an ActiveX control to display the DICOM data. The line below must be in any plug-in that has an API interface for compatibility with the Microsoft **IDispatchEx** interface:

```
<SCRIPT language="javascript" > </SCRIPT>
```

Failure to add this line will result in potential crashes and memory corruption problems.

3.2 Adding a Preference Page

Three sample **Preferences** pages were added by the “ISR Dicom Write” plug-in: User, System, and Machine. Although the three **Preferences** pages were added, the plug-in only makes use of the Machine preferences to set the **LogWriter** preferences. Study this sample to understand how to use a custom preference page with your plug-in. To view the Preference pages, navigate to **Preferences**, **User Preferences**, and **Sample ISR Plug-in** to view the sample.



☑ Enable Preferences

Preference One:

Preference Two:

Preference Three:

☐ Check value to store as preference

Technical Support: 1-(877)-328-2808 support@philips.com

The source sample for this example is:

```
[INSTALL DIR]\Samples\PlugInSamples\ISR PlugIn
Sample\ISRSamplePrefsUser.htm.
```

The sample was added by this line in DicomFile_ISR.htm:

```
function AddPrefPage()
{
// use the prefs file located in this same location
var thispath = document.location.toString();
thispath = thispath.toLowerCase();
var locate = thispath.indexOf("dicomfile_isr.htm");
if (locate > -1)
{
var path = thispath.substring(0, locate);
var prefpath = path + SAMPLE_PREFS_FILE;
Radiology.AddPreferencePage(SAMPLE_PREFS_NAME, prefpath,
SAMPLE_PREFS_TYPE);
}
}
```

3.2.1 Understanding Events for Preference Dialog Pages

Special attention should be paid to make sure you are handling events properly in your **Preferences** page. To enable the **Apply** button in your dialog box, you must detect a change, and enable the **Apply** button:

```
function SetModified(bFlag)
{
// If the user changes something we
//   need to enable the apply button
if (m_bModified != bFlag)
{
m_bModified = bFlag;
if (m_bModified == 1)
{
Radiology.EnablePreferenceApplyButton();
}
}
}
```

If the user presses **OK** or **Apply**, you will receive an **EventPreferencesApply** event. This event is where you check the input from the user to make sure it is correct, and you must also filter the event to make sure it is for your **Preference** page, because your plug-in will receive all events, even ones not generated by your preference dialog page. If it is your event, and the input is correct, only then should you write the Preferences:

```
function Radiology_EventPreferencesApply(PrefPageName, PrefType)
{
// only do this for our preferences
if ((PrefPageName == SAMPLE_PREFS_NAME) &&
(PrefType == SAMPLE_PREFS_TYPE ))
{
// no matter what happens we need to clear this flag.
SetModified(0);
SetPreferences(); // Calls Radiology.SetPreference()
}
}
```

When the Preferences are written to the database, they are updated immediately. If your plug-in must change based on the Preferences changing, use the **EventPreferencesApplied()** in your plug-in to alert you that the Preferences have changed.