iSite PACS Client and API Overview

iSite PACS 3.6 and 4.1

Rev. 1.03
2008 Dec 2

Philips Healthcare Informatics, Inc.
Enterprise Imaging

PHILIPS

PHILIPS

Enterprise Imaging

iSite PACS Client and API Overview

iSite PACS 3.6 and 4.1

Rev. 1.03
2008 Dec 2

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

Table of Figures

1 Introduction

This document provides an overview of the iSite PACS Client API, as well as implementation methods that should be used when developing any custom application using the iSite PACS API.

2 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 PACS:

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

3 iSite PACS Client Overview

3.1 iSite PACS User Interface

To correctly use the iSite PACS API, the application developer should have a solid conceptual understanding of how the iSite PACS client looks and behaves. Understanding how and where the API can be used to manipulate the iSite PACS Client will result in a better designed and more user-friendly custom iSite PACS application.

The Client mainly consists of two screens. The **Folder/Worklist** screen is used for performing functions such as finding exams, resolving exceptions, burning exams to CD's, and so on. The **Canvas Page** screen is used for viewing and manipulating exam images.

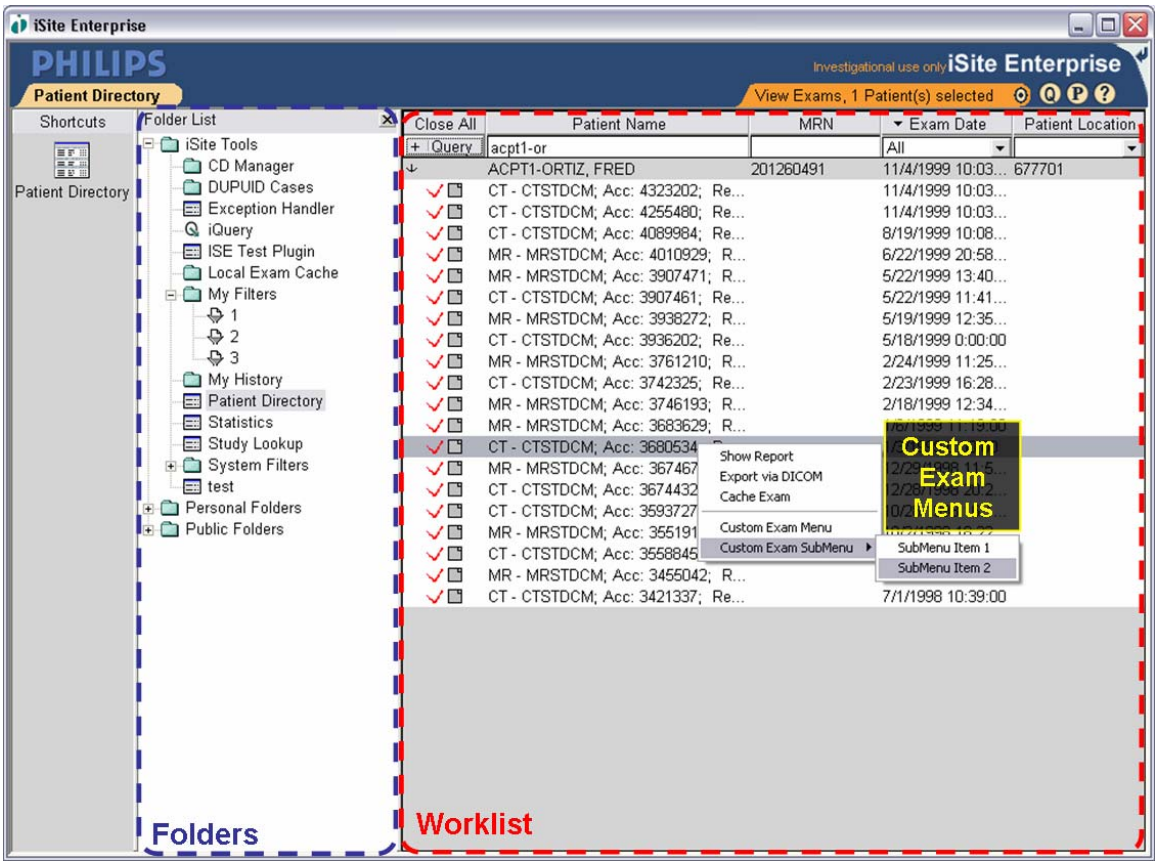

Figure 1: iSite Enterprise Folder/Worklist screen

The **Folders** panel is the navigation panel for the Folder/Worklist screen. The **Worklist** panel will change dynamically depending on the folder or tool selected in the **Folders** panel. Custom API plug-ins that are visible will display in the **Worklist** panel when they are selected by the user.

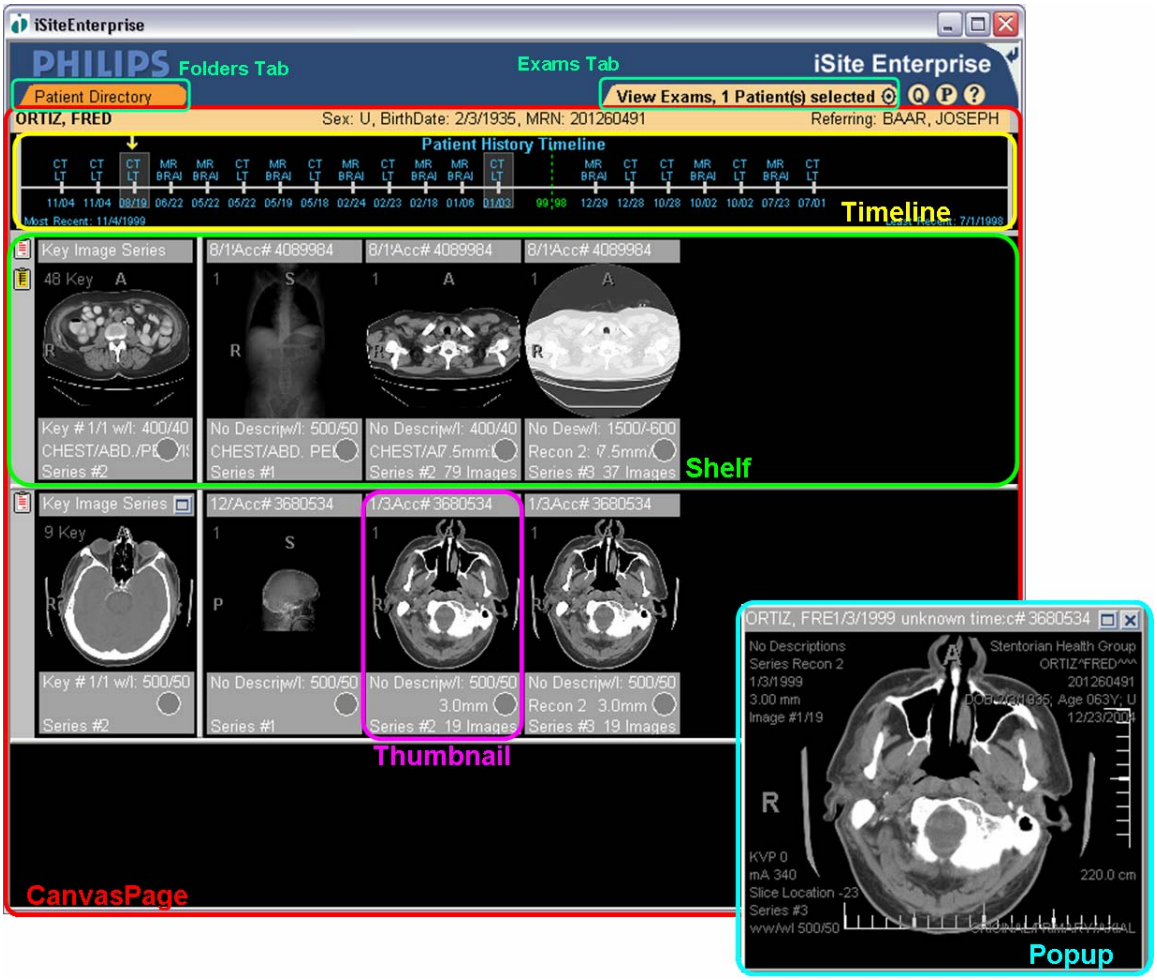

Figure 2: iSite PACS Canvas Page with a popup window

Canvas Page

- Main container for the patient image displays.
- More than one Canvas Page can be open at a time, however a Canvas Page is only associated with a single Patient ID.
- Contains a single timeline listing all exams for the patient.
- Always contains at least one shelf.
- Can contain multiple shelves.
- Each Canvas Page will have a unique **CanvasPageID**.

Timeline

- One timeline per Canvas Page.
- Shows patient exams in chronological order (most recent on the left).
- Right-click context menu can contain custom API menus (see Figure 3).

Shelf

- The shelf is the container for a single exam.
- Multiple shelves can be opened in a Canvas Page; however each shelf will be associated with a specific **ExamID**.
- Each shelf will contain at least one image thumbnail.
- Each shelf will have a unique **ShelfID**.

Thumbnail

- Each series [**DICOM SeriesUID**] in an exam will be contained in a separate thumbnail (or "stack") in a shelf.
- If an exam series contains multiple images, the images will be stacked in the thumbnail window.
- Each thumbnail will have a unique **WindowID**.

Popup Window

- A popup window is a free floating, resizable window.
- A popup contains a single exam series, similar to a thumbnail.
- Multiple popups can be open at the same time.
- Each popup window will have a unique **WindowID**.
- If the user navigates away from the Canvas Page associated with a popup without closing the Canvas Page, the popup window is hidden, not destroyed. When the Canvas Page is made visible again, the popup window will reappear.

Diagnostic Window

Note: This is available in Site Radiology only. iSite Enterprise does not have Diagnostic Windows.

- At least one Diagnostic Window is opened automatically for the first shelf loaded in a Canvas Page.
- If a client computer has multiple monitors enabled, a separate Diagnostic Monitor will be opened for each non-primary monitor.
- The Diagnostic Windows persists for as long as the Canvas Page persists.
- Each Diagnostic Window will have a unique **WindowID**.

Folders Tab and Exams Tab

- Tabs for navigating between the Canvas Page and the Folder/Worklist screen.

Note: **CanvasPageID's**, **ShelfID's**, and **WindowID's** are assigned dynamically at the time of creation and are valid from the time the object is created to the time the object is destroyed. iSite PACS creates a unique identifier for each image window it creates.

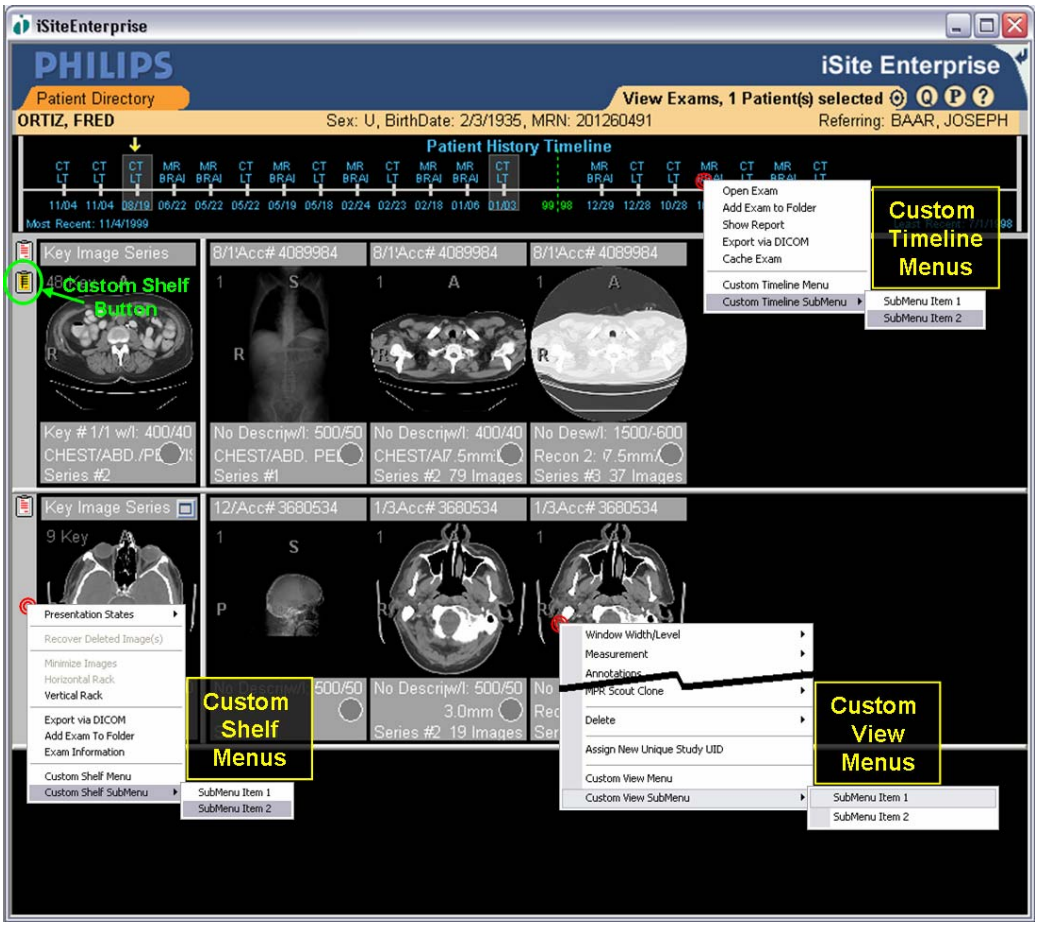

Figure 3: iSite PACS Canvas Page with API custom features

3.2 Clinical Data

Patient Data

The iSite PACS database manages patients using an internal private identifier. This identifier is used within iSite PACS to uniquely identify a single patient. Each unique patient is presented as a single tab in the control and as a single entry in the Patient Worklist. Managing patients using an internal ID allows iSite PACS to support multiple Patient IDs or MRNs per patient. This can occur in enterprises where multiple organizations identify patients differently. The API provides an easy way to convert an organization-specific patient identifier to the iSite PACS internal patient identifier. Please refer to the **FindPatient** function in the *ActiveX Reference Guides*.

Exam Data

The next level of organization granularity is the exam. iSite PACS exams present a RIS/HIS (Radiology Information System/Hospital Information System) view of the radiology department rather than the DICOM study based view. An exam has exactly one diagnostic report and may contain multiple DICOM studies. A study may belong to multiple exams. This relationship is necessary to support the following scenarios:

- A full body CT is ordered but billed and diagnosed as three separate exams (head, chest, abdomen).
- Mistakes are made at the modality that result in multiple DICOM studies for the same exam.

The iSite PACS database manages exams using an internal exam identifier. This identifier is used within iSite PACS to uniquely identify a single exam. Each unique exam is presented as a child entry in the Patient Worklist and as a single row, or shelf, on the Canvas Page. Managing exams using an internal ID allows iSite PACS to support enterprises where Accession numbers are not unique (both within an organization and between organizations). The API provides an easy way to convert an organization specific accession number to an internal exam identifier. Please refer to the **FindExam** function in the *ActiveX Reference Gu*ides.

3.3 iSite PACS Development Overview

3.3.1 iSite PACS Unified Client

Starting with iSite PACS version 4.1/3.6, the iSite Enterprise and iSite Radiology clients are unified and can be switched at the time of initialization to be able to communicate with either the traditional 3.x IDX Backend, or the 4.x Philips Backend. By default, the Clients run in IDX Backend mode and present 3.x version information. Specifying the StentorBackEnd option will initialize the Client in Philips Backend mode and present 4.x version information.

The APIs are the same in either mode. It is only necessary to know which type of backend you are developing for and make sure that the Option and ImageSuiteURL parameters are set correctly.

3.x Mode

Option = “” (Do not specify StentorBackEnd option)

ImageSuite = “http://<iSiteServer>/iSuite”

4.x Mode

Option = “StentorBackEnd”

ImageSuite = “http://<iSiteServer>/iSiteWeb/WorkList/PrimaryWorkList.ashx”

3.3.2 Client 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** or **Machine Preferences**, as follows:

- Plug-ins that are 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 that are added via the **System** preferences 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** checkbox is selected by default whenever a new plug-in is configured. You must select this check box 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. 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** 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. On the other hand, a plug-in that integrates a dictation application into the functioning of iSite PACS would be best suited for a **Machine** plug-in (because 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. Note the following:

- There is no inherent connection or communication between a plug-in and any custom preference page it adds.
- You should 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. Because 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.

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 **Disable API** 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 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 the following:

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

3.3.3 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 for a user to access the **Preferences** dialog box to disable the plug-in. For example, a new plug-in causes the iSite PACS client to crash immediately after login because of a bug in the onload functionality in the plug-in. 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.

3.4 iSite Radiology Development Overview

The iSite Radiology plug-in API is intended to be controlled with a web page scripting language, such as JScript, and is not restricted to be used with web pages. URLs containing scripting code are added through either a Plug-in Page via Preferences, or a BrowserPageURL via the iSite.ini file. Refer to the SDK documentation for “getting started” sample code.

3.4.1 iSite Radiology Design Considerations

It is important to keep these design considerations in mind while designing your web application.

- iSite Radiology is designed to be a diagnostic workstation. The application does assume a major role in the usage of system resources, such as memory, monitors, etc. If run as a standalone application it will take over the desktop and the high-resolution monitors. In integration mode, it will also take over the desktop but the worklists and rack can be shown or hidden by the controlling web page.
- iSite Radiology is designed to communicate with one server only. The server IP address is set in the iSite.ini file. There is no way to dynamically disconnect from one server and connect to a different server during runtime.

3.5 iSite Enterprise Development Overview

The iSite Enterprise application consists of the following ActiveX controls packaged as a single ocx:

- A non-visual control that allows the user to interact directly with the iSite PACS Server.
- A visual control that contains the user interface for the iSite Enterprise application.

- A single image control that contains a user interface for single image display.

Note: The iSite Enterprise control depends upon the non-visual control and will not operate until the non-visual control has been initialized. You must first initialize the non-visual control before initializing and using the **iSiteEnterprise** control.

3.5.1 iSite Enterprise Design Considerations

It is important to keep the following design considerations in mind while designing your web application:

- There can be only one instance of the **iSiteNonVisual** and **iSiteEnterprise** controls per process. Each execution of the Internet Explorer is a process. Note that if you open a popup window, (JavaScript – **window.open()**), it is still the same process even though the user interface looks like there is a second execution of the Internet Explorer.
- It is best if your main page or window instantiates the **iSiteNonVisual** control and the subsequent pages or dialogs instantiate the **iSiteEnterprise** control.

The **iSiteEnterprise::Initialize()** will find the **iSiteNonVisual** control on its own if the object passed in is a reference to the **iSiteEnterprise** itself. This is helpful because you do not have to pass a reference of the **iSiteNonVisual** to any subsequent pages and the **iSiteEnterprise** can function correctly. For example, you may instantiate the **iSiteNonVisual** in your main page and log in. Then in a frame instantiate the **iSiteEnterprise** and call **Initialize()** passing a reference to itself as the parameter.

```
iSite.Initialize(iSite)
```

This resolves the issue of having to pass the user and password across pages. It also allows the ability to create and destroy pages with the **iSiteEnterprise** control without having to logout and recreate the **iSiteNonVisual** control.

The **iSiteEnterprise** control requires a valid **iSiteNonVisual** control to be alive. The results are unpredictable and potentially fatal if you allow the **iSiteNonVisual** control to be destroyed while the **iSiteEnterprise** is still active.

3.5.2 iSite Enterprise Supported Configurations

You may use the controls in any manner, given that you stay within certain guidelines. These controls are more complex and dependent than a standard calendar control for example because they maintain login, state, and other communications with the iSite PACS Server.

The best approach to incorporating the controls is to instantiate, initialize, and keep available a non-visual control then to create and destroy an **iSiteEnterprise** control at will. Creating the non-visual control takes a little time because the control will communicate with the server to initialize itself and download any preferences, and so on.

Applications (C++, VB)

VB, MFC, MDI applications, etc. are acceptable as long as there is only one non-visual control instantiate per process and that the non-visual is created and initialized before the **iSiteEnterprise** control is initialized. You may destroy all instantiations of all iSite PACS controls and recreate them, but you are not allowed to destroy the non-visual while there is still an **iSiteEnterprise** control alive. The existing **iSiteEnterprise** control will not bind to the new non-visual control.

Web Based Applications running Internet Explorer 5.0 or later

In single Internet Explorer window implementations, style sheets, frames, and dialogs boxes are fine as long as the non-visual control does not get destroyed. See *Unsupported Configurations* below for Internet Explorer popup implementation issues.

3.5.3 iSite Enterprise Unsupported Configurations

Multiple Instances of the Controls in Same Process

The iSite Enterprise controls are more complex than the standard web based controls in that they establish and maintain communications with a server. The two controls also maintain communication with each other. The **iSiteEnterprise** control uses the non-visual control for certain communications with the server. Therefore these two controls must be used properly. There cannot be two instances of either control in the same process at the same time. It is also incorrect to destroy the non-visual control while there is still an active **iSiteEnterprise** control. Please note that in Internet Explorer implementations that popup windows are still at part of the same process but are running in a different user interface thread. See the next section for details.

Internet Explorer Popups

iSite Enterprise only supports single instances of the controls per process. Each running application is a process. Through scripting a developer may place a popup window on the screen. This popup window looks and acts like a separate instance of the Internet Explorer but is actually in the same process running a different user interface thread. Because of the internal design of the controls we cannot support running the controls in a user interface thread other than the primary user interface thread. Therefore, iSite Enterprise cannot support running in Internet Explorer popups at this time.

3.5.4 Common Pitfalls

Login Issues

Logging in is an issue for web-based applications because the user and password must be available for the non-visual control. If the non-visual control is not on the same web page as the login screen, there must be a secure way to get the user and password to the non-visual control. This issue should be considered in designing your web-based application.

Duplicate Instantiations of the Controls

Only one instance each of the **iSiteNonVisual** and **iSiteEnterprise** controls are allowed per process. Each instance of your application or each launch of the Internet Explorer is a separate process. Note that an Internet Explorer popup window is still running in the same process as the parent Internet Explorer window even though it looks like a new instance of the browser. Care should be taken in your design that you don’t create two instances of the controls at the same time and that you don’t delete a non-visual control that an iSite Enterprise control is using. If you do this, the results are unpredictable.

Instantiating Both Controls at Display Time

If you design your application to instantiate both controls at the same time just before displaying each image and destroying them both when done will lead to slow performance. The **iSiteNonVisual** control has to connect to the server and download Preferences and log

in before an image can be displayed. This latency can be overcome by instantiating and initializing the non-visual control at startup only, then instantiating and destroying only the **iSiteEnterprise** control at display time.

Setting Options After Initialization

The options are read by the **iSiteEnterprise** control during initialization. Options changed after initialization will not go into effect. Initializing a second time will not reset the options.

3.5.5 Requirements

Use and implementation of the iSite PACS SDK requires a Philips NDA (Nondisclosure Agreement). Your integration project needs to be discussed with Philips and the integration design approved by Philips Radiology Informatics Customer Engineering. The overall design of your project does not need approval, just the areas that affect iSite PACS integration. An additional iSite PACS Server is required to be deployed for development and upgrade purposes. This server will not be used as a production server. Finally, the upgrade plans need to be discussed and agreed upon.

iSite PACS System requirements:

Internet Explorer 5.0 or later (Netscape is not supported).

3.5.6 Visual Basic Notes

The iSite PACS ActiveX controls will cause Visual Basic to crash when changing between design mode and run mode. To avoid this, set the DWORD registry key `HKEY_LOCAL_MACHINE\Software\Stentor\VBMode` to 1 and restart Visual Basic.

3.6 iSite Enterprise Quick Start

3.6.1 Adding iSiteEnterprise Controls to a Web Page

Remember that there can only be one **iSiteNonVisual** control per process.

It is best to make the **iSiteNonVisual** control not visible by setting the size to zero. You may alternately set the display attribute of the object's style to "none." Refer to the documentation on style sheets for more information.

If you use a visual environment you may graphically add the controls to your web page. Otherwise you can use the code provide here. Place this object in the body of your web page. Note the width and height parameters are set to 0%. This will prevent any visual display of the **iSiteNonVisual** control.

3.6.2 Using the iSite Enterprise Control

The following HTML sample shows how to use the iSite Enterprise controls in your web page. This example instantiates and initializes the two controls.

```
<HTML>
<HEAD>
<title>Philips iSite Enterprise</title>
</head>
<body scroll="no" leftmargin=0 topmargin=0 BOTTOMMARGIN=0 RIGHTMARGIN=0
onload="window.status= '&copy; 2000 - 2006 Koninklijke Philips Electronics
N.V.'">

<OBJECT id=iSiteNonVisual
 CLASSID = "clsid: ISITE NON-VISUAL CLSID"
 width=0%
 height=0%>
 <PARAM NAME=iSyntaxServerIP VALUE="eng-mp10">
 <PARAM NAME=ImageSuiteURL VALUE="http://eng-
mp10/iSiteWeb/WorkList/PrimaryWorkList.ashx">
 <PARAM NAME=ImageSuiteDSN VALUE="iSite">
 <PARAM NAME=Options VALUE="StentorBackEnd">
</OBJECT>

<OBJECT id=iSite
 CLASSID="clsid: ISITE VISUAL CLSID "
 width=100%
 height=100%>
 <PARAM NAME=Options VALUE="StentorBackEnd">
</OBJECT>

<SCRIPT Language = VBScript>
On Error Resume Next
iSiteNonVisual.Initialize()
iSite.Initialize(iSiteNonVisual)
</SCRIPT>

</body>
</HTML>
```

3.6.3 Using the iSite Image Control

When using the iSite PACS database through the single image control, individual images are accessed based upon the Exam, Study, and Image UIDs. Please refer to the **DisplayImage** or **DisplayStack** functions in the *iSite Enterprise ActiveX Reference Guide*. All these identifiers are external identifiers and must be known before any operation can be performed.

After an image or a stack of images has been displayed, several methods can be used to obtain information about the currently displayed image. Refer to the **GetWindowContext** or **GetStaticWindowInfo** functions for more information.

The following HTML sample shows how to use the iSite Image control in your web page. This example instantiates and initializes the two controls.

```
<HTML>
<HEAD>
<title>Philips iSite Image</title>
</head>
<body scroll="no" leftmargin=0 topmargin=0 BOTTOMMARGIN=0 RIGHTMARGIN=0
onload="window.status= '&copy; 2000 - 2006 Koninklijke Philips Electronics
N.V.'">

<OBJECT id=iSiteNonVisual
 CLASSID = "clsid: ISITE NON-VISUAL CLSID"
 width=0%
 height=0%>
 <PARAM NAME=iSyntaxServerIP VALUE="eng-mp10">
 <PARAM NAME=ImageSuiteURL VALUE="http://eng-
mp10/iSiteWeb/WorkList/PrimaryWorkList.ashx">
 <PARAM NAME=ImageSuiteDSN VALUE="iSite">
 <PARAM NAME=Options VALUE="StentorBackEnd">
</OBJECT>

<OBJECT id=iSiteImage
 CLASSID="clsid: ISITE IMAGE CLSID "
 width=100%
 height=100%>
 <PARAM NAME=Options VALUE="StentorBackEnd">
</OBJECT>

<SCRIPT Language = VBScript>
On Error Resume Next
iSiteNonVisual.Initialize()
iSiteImage.Initialize(iSiteNonVisual)
</SCRIPT>
</body>
</HTML>
```

3.6.4 Developing a Plug-In for Both iSite Radiology and iSite Enterprise

It is possible to develop the same plug-in for both iSite Radiology and iSite Enterprise by setting the client type in the plug-in OnLoad event. For example, the user could do the following:

```
if (typeof(iSiteEnterprise) == "object") {
    iSite = iSiteEnterprise;
} else if (typeof(Radiology) == "object") {
    iSite = Radiology;
} else {
    iSite = NotDefined;
}
```

Note: The response of the API function calls for both iSite Radiology and iSite Enterprise should be verified using the corresponding Reference Guides.