Service Manual

PN 0889579

04/08/10

RadWhere 3.0

PACS Integration Installation/Configuration

IMPORTANT

The installation/configuration procedures in this manual should be performed only by thoroughly trained and knowledgeable Nuance Personnel.

Table of Contents

Overview

RadWhere can be integrated with other third party applications (e.g., PACS) on the Workstation desktop. This section lists and describes all of the types of integrations that are currently supported. Included are instructions for configuring RadWhere for these integrations using the Portal Web administration application. Also provided is information about the RadWhere SDK and the various APIs and tools that are available to the application developer.

The main purpose of the desktop integration is to supply automatic context sharing between the vendor application (PACS, RIS, etc.) and RadWhere. This context includes study information, such as accession numbers, patient identification, and end-user information in the form of logon ids, passwords, etc. In some cases, it also provides the ability to share image and scene information between RadWhere and certain PACS applications. Finally, the low level APIs provide more detailed workflow event notifications to the integrated vendor application, such as when a report is opened or closed, a user logs off, or RadWhere is terminated.

Types of Integrations

There are two basic types of RadWhere desktop integrations:

Slave Mode - In Slave mode, the end user is using the PACS system, or other desktop application, to manage the work list of patients and studies. RadWhere acts as a **slave**, synchronizing its work list to the same patient and study context as the Master.

Master Mode - In Master mode, the end user is using the RadWhere work list, and it is up to RadWhere to notify the PACS system of changes in context.

Supported Integrations Options

Name	Description	Master/ Slave
RadWhere Standard Integrations		
IRadWhere	This is the .Net Remoting interface supported directly by RadWhere. This interface is the base interface used by all other desktop integrations. This interface can only be used with a Microsoft .Net application (i.e., c#, vb.net, etc.). It is currently not used directly by any RIS/PACS vendor. ● Single sign-on ● Logout and terminate ● Accession number context sharing ● Events for all RadWhere workflow actions	Slave
RadWhereCOM RadWhereAx	This **API** form of integration is a COM component that "wraps" all of the functionality of the base remoting interface (IRadWhere). This component can be used by non-.Net languages that support COM, such as VB6 and Delphi. RadWhereAx is an ActiveX wrapper of RadWhereCOM. ActiveX is needed for web integrations, such as Slave Mode Philips iSiteRadiology. Features: ● Launch ● Single sign-on ● Logout and terminate ● Accession number context sharing ● Events for all RadWhere workflow actions (e.g., signed/discarded reports)	Slave
RadWhereActiveX (e.g., Philips iSite Master Mode)	This integration was originally implemented to support the iSite Web Plug-in for Master Mode. Theoretically, it could be used for other Master Mode integrations. Accession number context sharing is the only feature currently provided.	Master
RWCommand	RWCommand is a command line interface. This is used for integrations where the controlling application (e.g., PACS) can communicate using a command line ● Single sign-on (3.0 only) ● Accession number context sharing ● RWCommand also supports Slave Mode integration with AquariusNET that allows the PACS to specify image and scene attachments ● RWCommand is currently used in conjunction with the following PACS integrations: GE Centricity, AquariousNET, and Amicas.	Slave
RadWhere XML	This **RadWhere standard** integration is implemented using an XML file drop interface. ● Single sign-on ● Accession number context sharing	Slave
RadWhere URL	This **RadWhere standard** integration is implemented using a URL command line. ● Accession number context sharing	Master

Name (Continued)	Description (Continued)	Master/ Slave
TCP/IP Socket	This integration uses TCP/IP socket communication to share accession number context only. ● Accession number context sharing	Master
Proprietary Integrations		
PowerScribe Two-way XML Integration	This bidirectional XML file interface is used in the Hologic SecurView integration and later version of AGFA Impax. It can be used for any integration in both Master and/or Slave Mode. **Note:** Two distinct PACS connectors must be created in the Portal Sites page to support both Master and Slave. ● Single sign-on ● Logout and terminate ● Accession number context sharing ● Events for all RadWhere report status changes	Both
TeraRecon AquariusNET	Master Mode integration is provided using an AquariusNET executable (AquariusNetViewer.exe) that RadWhere executes from the command line. This allows RadWhere to launch images and scenes. **Note:** Slave Mode integration is provided by an XML file drop used in conjunction with **RWCommand** (see above).	Master
Epic Radiant Master Mode	Master Mode integration is implemented by RadWhere making calls using the Epic COM component per the Hyperspace PACS Integration Specification. This integration supports user, password, and accession number context sharing. **Note**: Slave Mode integration with Epic Radiant is currently done using the RadWhereCOM component to integrate with RadWhere. See **RadWhereCOM** above. ● Launch ● Single sign-on ● Logout and terminate ● Accession number context sharing ● Events for all RadWhere workflow actions	Master
McKesson/ALI Horizon	This integration is implemented using two-way XML file communication that follows the McKesson specification. Besides accession number context sharing, it also allows RadWhere to acknowledge receipt of accessions. It will also notify Horizon when a report is closed and provide the report status. ● Accession number context sharing ● RadWhere report status change event notification	Slave
Siemens MagicView	This integration is a Master Mode two-way file drop implementation written to the Siemens specification. MagicView will share accession context with RadWhere, and RadWhere will write a response file indicating failure or success. ● Accession number context sharing	Master

Name (Continued)	**Description (Continued)**	**Master/ Slave**
Mercury/Visage	Master Mode integration is provided using an executable (oemquery.exe) that RadWhere executes from the command line. This allows RadWhere to launch images using accession number context sharing. ● Accession number context sharing	Master
Siemens Syngo (legacy)	This is a Slave Mode-only interface that used a one-way XML file drop. ● Accession number context sharing	Slave
VitalConnect	This Master Mode integration supports the viewing of images based on accession number. User and password context sharing is supported also. ● Accession number context sharing ● Single sign-on	Master
Intelerad	This is a COM API used for master mode only. It will launch the Intelerad viewer (InteleViewer) and load images by accession number and MRN.	Master
GE Centricity	This integration uses the GE COM API. It is used for Master Mode only. ● Application launch ● Accession number/MRN context sharing	Master
GE RIS - ImageCast	this is a Slave Mode integration that allows RadWhere to be driven by the RIS-IC work list. ● Application launch ● Single sign-on ● Accession number/MRN context sharing ● Report status change notification	Slave

Installation and Configuration

Configure PACS in Portal

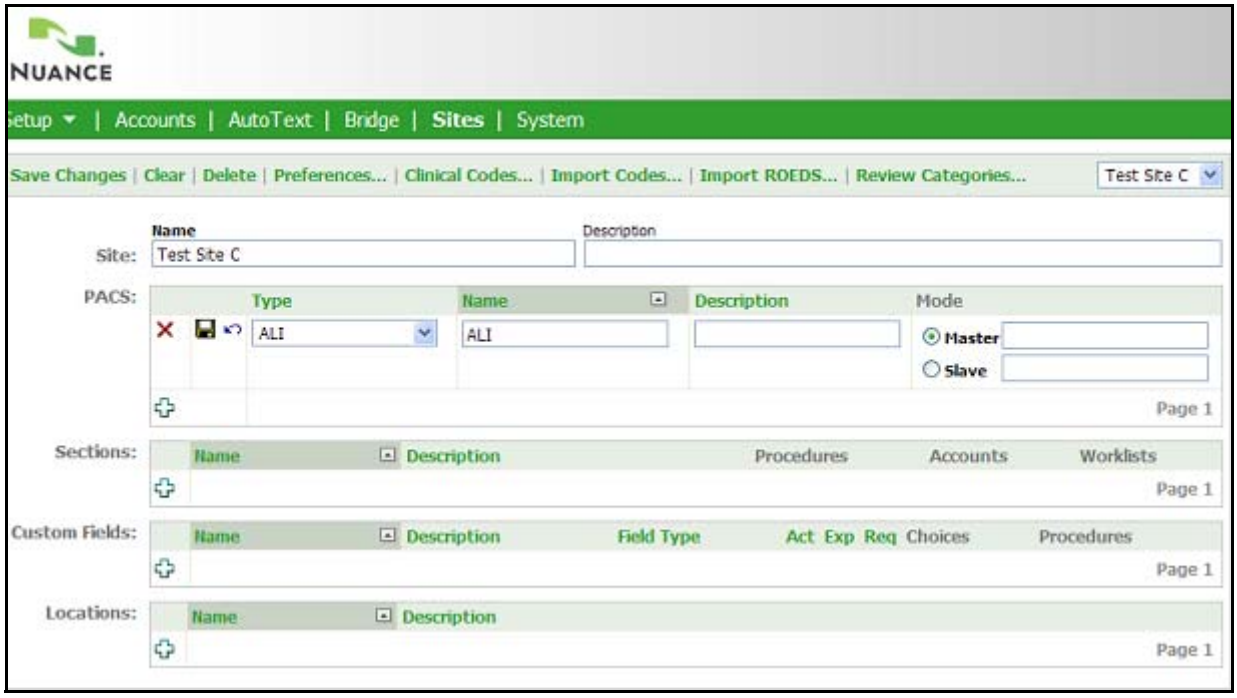
Most PACS integrations offered by RadWhere can be configured in the Portal. Exceptions to this are listed in subsequent sections.

1. To configure PACS in Portal, open the Portal **Sites** page, **http://localhost/Portal/admin/sites.aspx**.

   The Web page will contain a control for **PACS** in addition to other items, such as Section and Work lists.

2. To add a new PACS configuration, perform the following:

   a. Click **+** (plus sign) in the PACS control to add a new entry.



   b. From the drop-down, choose the Integration type.

   c. Enter the Site values for **Name** and **Description** as appropriate (e.g., Syngo for Name, Syngo Boomerang XML Integration for description).

   d. Choose **Slave** Mode or **Master** Mode.

   e. Enter the options as needed. See section Configure Master and Slave Options below for details configuring the options for specific PACS types.

Configure Master and Slave Options

PACS Type	Options
RadWhereActiveX (Master)	Host name of PACS client. This is used to create a URL that is used to communicate the accession number. Currently, this value **must** be specified as **http://localhost:8080**.
RadWhere XML (Slave)	There are two options that can be specified. The first one is required: 1. **(Required)** Complete file path of the XML file dropped to RadWhere from the PACS. Example: **c:\Nuance\xml_in\study.xml** 2. **(Optional)** Multi-site - determines how RadWhere will search for the order(s) specified by the PACS: ● **multisite=all** (Default) All sites will be searched ● **multisite=thissite** Only the configured site will be searched ● **multisite=selected** Search is based on the value selected in site drop-down box in RadWhere 3. **(Optional)** Encryption key - If single sign-on will be used, this is the password used by the RadWhere encryption algorithm. The same password is used by the PACS when encrypting the user name and password for single sign-on. **Since this is an optional parameter, it must be specified with *parameter*=** (e.g., parameter=12345). If this PACS type is being used in conjunction with RWCommand, as in the GE Centricity scenario, this value must match the value specified by the Administrator when installing the RadWhere Components from install.bat at the initial installation of each workstation. Example: **c:\Nuancexml_in\study.xml;password=12345**
RadWhere URL (Master)	Formatted URL, using {accession}, {mrn}, {username} and {password} for substitution variables. Example: **http://PACS1/images?accession={accession}**
TeraRecon AquariusNET (Master)	No options required.
GE Centricity (Master)	No options required.
Dynamic Imaging (Slave)	1. **(Required)** TCP/IP Port Number - Port number on which RadWhere will be listening. 2. **(Optional)** Multi-site - Determines how RadWhere will search for the order(s) specified by the PACS: ● **multisite=all** (Default) All sites will be searched ● **multisite=thissite** Only the configured site will be searched ● **multisite=selected** Search is based on the value selected in site drop-down box in RadWhere Example: **9900;multisite=thissite**

PACS Type	Options (Continued)
McKesson Horizon (Slave)	1. **(Required)** Directory where XML files will be read and written by RadWhere. 2. **(Optional)** Multi-site - determines how RadWhere will search for the order(s) specified by the PACS: ● **multisite=all** (Default) All sites will be searched ● **multisite=thissite** Only the configured site will be searched ● **multisite=selected** Search is based on the value selected in site drop-down box in RadWhere Example: **c:\horizonxml;multisite=thissite**
Siemens MagicView (Master)	1. Directory where RadWhere will drop files to the PACS. 2 Directory where RadWhere will receive the response files from the PACS. **Note:** Only the directory names should be configured. The file names are automatically created by RadWhere based on the accession number. Example: **c:\MV1000\study;c:\MV1000\response**
Mercury/Visage (Master)	No options required.
RadNet (Slave)	1. **(Required)** Complete file patch of the XML file dropped to RadWhere from the PACS. Example: **c:\Nuance\xml_in\study.xml** 2. **(Optional)** Multi-site - determines how RadWhere will search for the order(s) specified by the PACS: ● **multisite=all** (Default) All sites will be searched ● **multisite=thissite** Only the configured site will be searched ● **multisite=selected** Search is based on the value selected in site drop-down box in RadWhere Example: **c:\radnet\study.xml;multisite=selected**

PACS Type	Options (Continued)
Powerscribe XML (Slave/Master)	**Slave**: Multiple parameters are specified for this integration. Each parameter is separated by a semicolon (;). Optional parameters must be preceded with a parameter name and the equal (=) sign. Example: **password=12345** 1. **(Required)** Input file path **-** Complete file path of the **study** file dropped to RadWhere from the PACS. **This parameter is positional and must be *first***. Example: **c:\Nuance\xml_in\study.xm** 2. **(Required)** Output file path - Complete file path of the **status** file dropped to the PACS. **This parameter is positional and must be *second*.** Example: **c:\Nuance\xml_out\status.xml** 3. **(Optional)** password - This is the Encryption password used by the RadWhere encryption algorithm. This value must be agreed upon by RadWhere and the PACS application when encrypting the user name and password. If single sign-on is used, this parameter is required. There is no default. 4. **(Optional)** base64 - This Boolean flag, **true** or **false**, is specified to indicate whether or not the encrypted user name and password are encoded using base64 encoding. The default is **true**. 5. **(Optional)** FieldName **-** This string is used to override the XML tag name of the element containing the accession number. By default, per specification, the name is **Field6**. 6. **(Optional)** MultiAccSeparator **-** This character is used to delimit multiple accession numbers for inbound and outbound XML files. The default is a comma (,). 7. **(Optional)** Multi-site - determines how RadWhere will search for the order(s) specified by the PACS: ● **multisite=all** (Default) All sites will be searched ● **multisite=thissite** Only the configured site will be searched ● **multisite=selected** Search is based on the value selected in site drop-down box in RadWhere **Example**: c:\Nuance\xml_in\study.xml; c:\Nuance\xml_out\status.xml; password=12345;base64=true;FieldName=AccessionNumber;MultiAccSeperator=\ ; multisite=thissite **Master**: Complete file path of the **status** file dropped to the PACS. Example: **c:\Nuance\xml_out\status.xml**

PACS Type	Options (Continued)
Fuji Synapse (Slave)	1. **(Required)** TCP/IP port number **-** The port on which RadWhere will be listening. 2. **(Optional)** Multi-site - Determines how RadWhere will search for the order(s) specified by the PACS: ● **multisite=all** (Default) All sites will be searched ● **multisite=thissite** Only the configured site will be searched ● **multisite=selected** Search is based on the value selected in site drop-down box in RadWhere Example: **9900;multisite=thissite**
Siemens Syngo Legacy (Slave)	1. **(Required)** Fully qualified path (including name) of the Siemens boomerang XML file. 2. **(Optional)** Multi-site - Determines how RadWhere will search for the order(s) specified by the PACS: ● **multisite=all** (Default) All sites will be searched ● **multisite=thissite** Only the configured site will be searched ● **multisite=selected** Search is based on the value selected in site drop-down box in RadWhere Example: **c:\syngo\boomerangxmlfile.xml;multisite=thissite**
VitalConnect (Master)	Base URL of VitalConnect web page.
Vitrea VxNet (Master)	No options required.
Intelerad (Master)	Four parameters are required for configuration: ● Intelerad's server URL ● Inteleviewer password ● Inteleviewer fallback user ● Authentication token All of the values must be provided by Intelerad or the Customer. Example: https://integration.intelerad.com;nuance;nuancefallback;0d855d6732104911c9a6712d90a2cc3c

Set the RadWhere Remoting Port

Most PACS integrations will require that a TCP/IP port be configured for the integration module to communicate with the IRadWhere .Net Remoting base integration. This port is usually set to **9090**, but it can be any available port desired. By default, this port is deactivated (i.e., set to zero), so it is imperative that you change it to a valid value.

This setting is configured in the Portal.

1. Open the Portal **Sites** page, i.e. http://localhost/Portal/admin/system.aspx.
2. Click **Configuration**.

   This will display a page where you can specify the remoting port in the **RadWhere Client** section:

RadWhere Connector Components

Installation

Desktop integration components are installed during the initial RadWhere Client installation. These components are installed by the RadWhereConnectorSetup install script, which is launched by the install.bat batch file. When the script is run, you will be prompted to enter the values in the following table.

Setting	Description	Registry Location
RemotingPort	The IP port that RadWhereCOM uses to communicate to RadWhere	HKLM\Software\Nuance\RadWhereCOM
RemotingHost	The host name that RadWhere is using	HKLM\Software\Nuance\RadWhereCOM
EncryptionKey	The password used by RWCommand and DPPlugin to encrypt user names and passwords. This value must match the password specified in RWCommand slave mode options.	HKLM\Software\Nuance\RWCommand

No further steps are needed for installation. The following sections, RadWhereCOM, RadWhereAx, and RWCommand, are presented for the purpose of information only.

RadWhereCOM

The COM component is used to integrate with certain systems, such as Sectra PACS and Epic RIS. It is also a prerequisite for RadWhereAx, covered below, which is used by Philips iSite.

When this component is installed, the installer is prompted to enter the RadWhere remoting port. The value entered must match the value entered in the Portal system configuration, which should have been configured as documented above.

RadWhereAx

RadWhereAx is an ActiveX control used to integrate to systems such as iSite (Philips/Stentor PACS), which require Web browser integration.

RWCommand

RWCommand is a Windows console application used for certain PACS integrations, such as GE Centricity. The installer for RWCommand.exe is available on the RadWhere FTP download

For these integrations, you must install RadWhere Connection Components on each Client Workstation.

PACS Specific Implementation Notes

This section describes the details involved in setting up specific PACS integrations. Contact Nuance Technical Support for additional information.

GE Centricity Slave Mode

This integration works by setting up new UI controls (i.e., buttons) in Centricity. These buttons are configured to execute a RadWhere command line program (RWCommand.exe), which in turn creates an XML file that is input to RadWhere. The XML file follows the RadWhere XML file specification.

Perform the following steps to set up the Centricity integration.

1. Obtain the GE *Generic Extend API Specification* document. This document applies to Centricity Versions 1.0-3.0.

IMPORTANT

*This step **must** be performed by a Site Administrator or GE Support Personnel. It requires Administrative access to the GE Workstation and its configuration files.*

2. Referring to Section 8 of the *Generic Extend API Specification* document, install the new buttons in Centricity. The buttons can be installed in any or all of the following three locations:

   - Extend drop-down menu on Work Modes palette
   - Exam functions drop-down menu in viewing
   - Image functions drop-down menu in viewing

   In each case, set the label to **Nuance** and the button label to **Start Nuance**. The ***text*** can be anything you choose (e.g., RadWhere, Start RadWhere), but the ***button label*** is the one that actually launches RadWhere.

3. Create a folder that will be used to drop an XML file to RadWhere (e.g., d:\Nuance).

4. Configure the button command property so that it executes RWCommand.exe:

   ...work list.component1.command=c:\Program Files\Nuance\Nuance RWCommand\rwcommand.exe -f D:\Nuance\order.xml -a <ACCN0>

   Optional: If single sign-on is desired, the **–u** and **–p** (i.e., user name and password) may be specified.

5. Configure the Centricity integration on Nuance Portal.
   a. Select the **RadWhere XML/URL** PACS type.
   b. Set the flags to **2** for **Slave** mode.
   c. The slave arguments should be set with the following values separated by a comma (,) character:
      - **The fully qualified path of where the XML file will be created.** This location much match the path specified when configuring the button command property. In the previous example, this was **d:\Nuance\order.xml**.
      - **(Optional) The encryption key used to encrypt and decrypt the password.** This value must match the value specified in the registry **HKLM\Software\Nuance\RWCommand\EncryptionKey**. This value is set when installing the RadWhere Components from install.bat at the initial installation of each Workstation.

        Example: **d:\Nuance\order.xml;pw123**

Philips iSiteRadiology

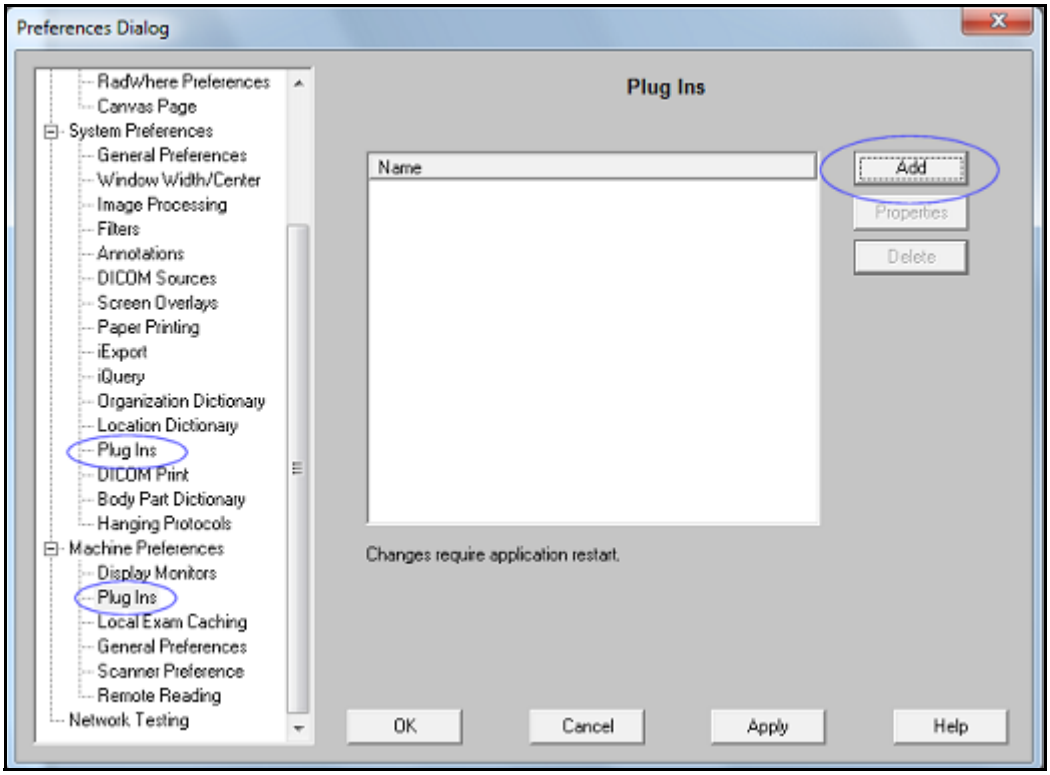
Philips (formerly Stentor) iSiteRadiology integration is achieved through a **Web plug-in**. A Web page developed by Nuance is configured in Philips as a plug-in. This plug-in will have access to the iSiteRadiology API using JavaScript code. It will also have an embedded ActiveX control (RadWhereAx) that is used by the plug-in to communicate with RadWhere. The ActiveX control listens for HTTP requests from RadWhere (Master Mode), and it uses the RadWhereAx control for Slave Mode.

Installation and Configuration

Perform the following steps to set up RadWhere for iSiteRadiology integration.

1. Log on to iSite Radiology and open the Preferences dialog by clicking the **P** icon in the upper right corner of the window.
2. Install the plug-in.

   The plug-in can be installed as a **system** plug-in or a **machine** plug-in.



   a. **System** plug-ins are installed ***once*** for ***all*** Workstations using the iSite server.

   b. If you do not want the plug-in available on all of the Workstations, you must install the plug-in as a **Machine** plug-in ***for each of the Workstations*** that need access to the RadWhere integration

3. After making the choice, either System or Machine plug-in, click **Add**.

The following dialog box displays.

4. Check the **Enable in iSiteRadiology** option.
5. Uncheck the **Disable API** option.

   The plug-in is located with the RadWhere Portal Web site in a subfolder named **pacs/isite**. In the example dialog box above, the Portal host name is **RWSERVER**.
6. Click **OK** to add this plug-in.

   You are returned to the previous dialog box.
7. Click **OK**.

The following step is **Optional** and is **only** for **Master Mode**.

8. **Master mode only**: Navigate to the Sites page in Portal to create a new PACS.
   a. Specify PACS type as **RadWhere ActiveX**.
   b. Specify **master** mode.
   c. Enter **http://localhost:8080** for the master mode argument.

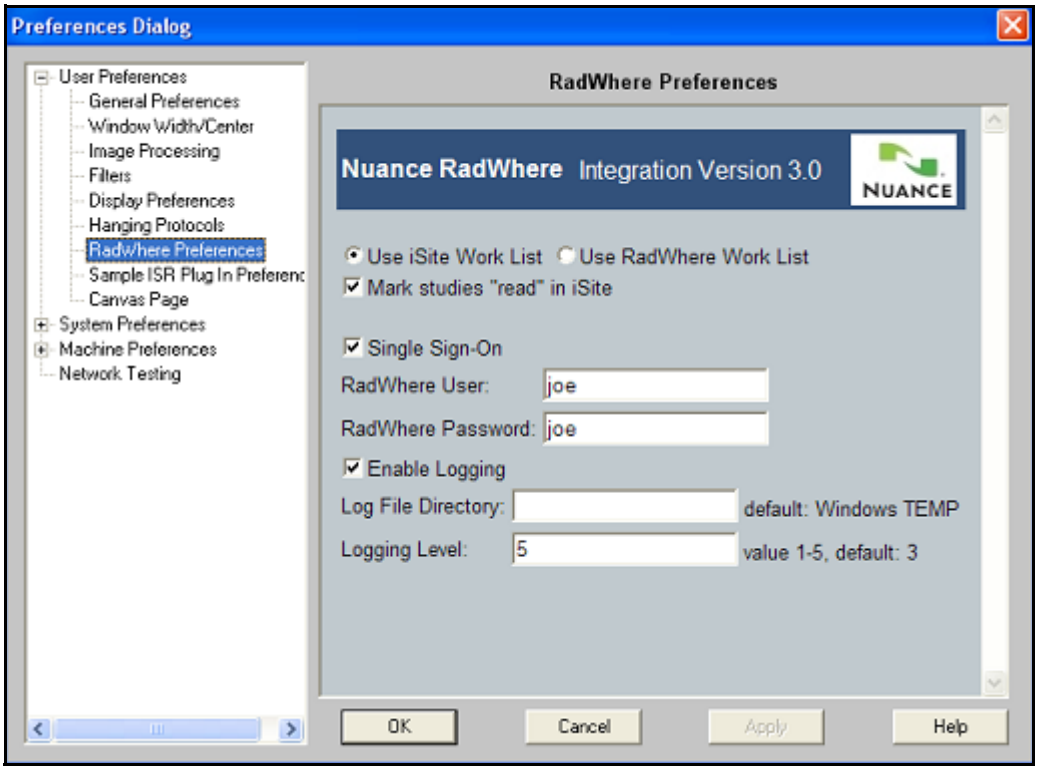
IMPORTANT

For both Master and Slave Modes: RadWhereCOM is required for this integration and is installed as part of the RadWhereConnectorSetup. *The remoting port must be set as described in the* *Set the RadWhere Remoting Port section on page 13* *and the* *RadWhere Connector Components section on page 14**. Windows Firewall blocks all TCP/IP ports, so make sure you open the configured port using the Windows Firewall setup utility.*

User Setup

1. For each iSite user, log on to iSite and go to the **Preferences Dialog** by clicking on the **P** icon in the upper right of the main window.
2. Under **User Preferences**, select **RadWhere Preferences**.

   The following dialog window displays.



3. Select the **Use iSite Work List** radio button to enable **Slave** mode ***or*** choose **Use RadWhere Work List** to enable **Master** mode.
4. Checking the **Mark studies read in iSite** option will cause studies to be marked as **read** (dictated) in iSite whenever a report has been saved or signed. When a report is discarded, the study is marked **unread** in iSite.
5. Choose the single sign-on option to automatically launch and login to RadWhere upon logging into iSite.

   You must specify a valid RadWhere user name and password.
6. The **Enable Logging** option is recommended.

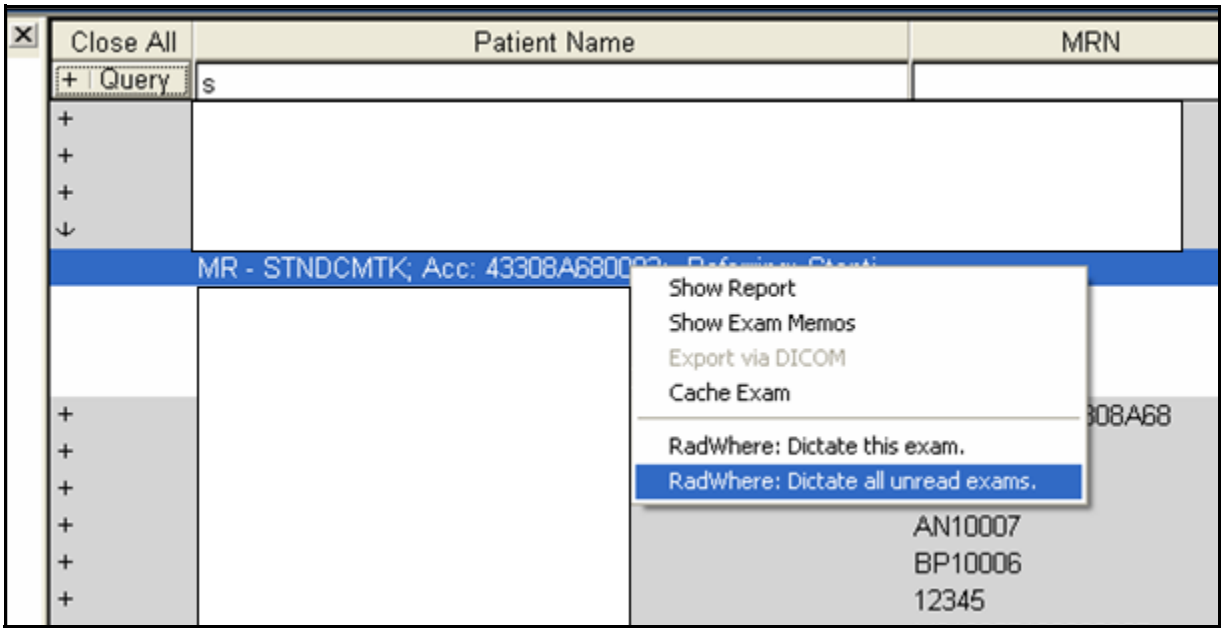
   Specify a log file directory, or leave it blank to store the log file in the user temp file directory (e.g., \documents and settings\<user name>\local settings\temp). Specify a logging level between 1-5. The higher the number, the more verbose the log file is.

Using the iSite – RadWhere Slave Mode Integration

For **Slave** mode operation, the users will have two **Dictate** menu items from which they can launch studies in RadWhere.

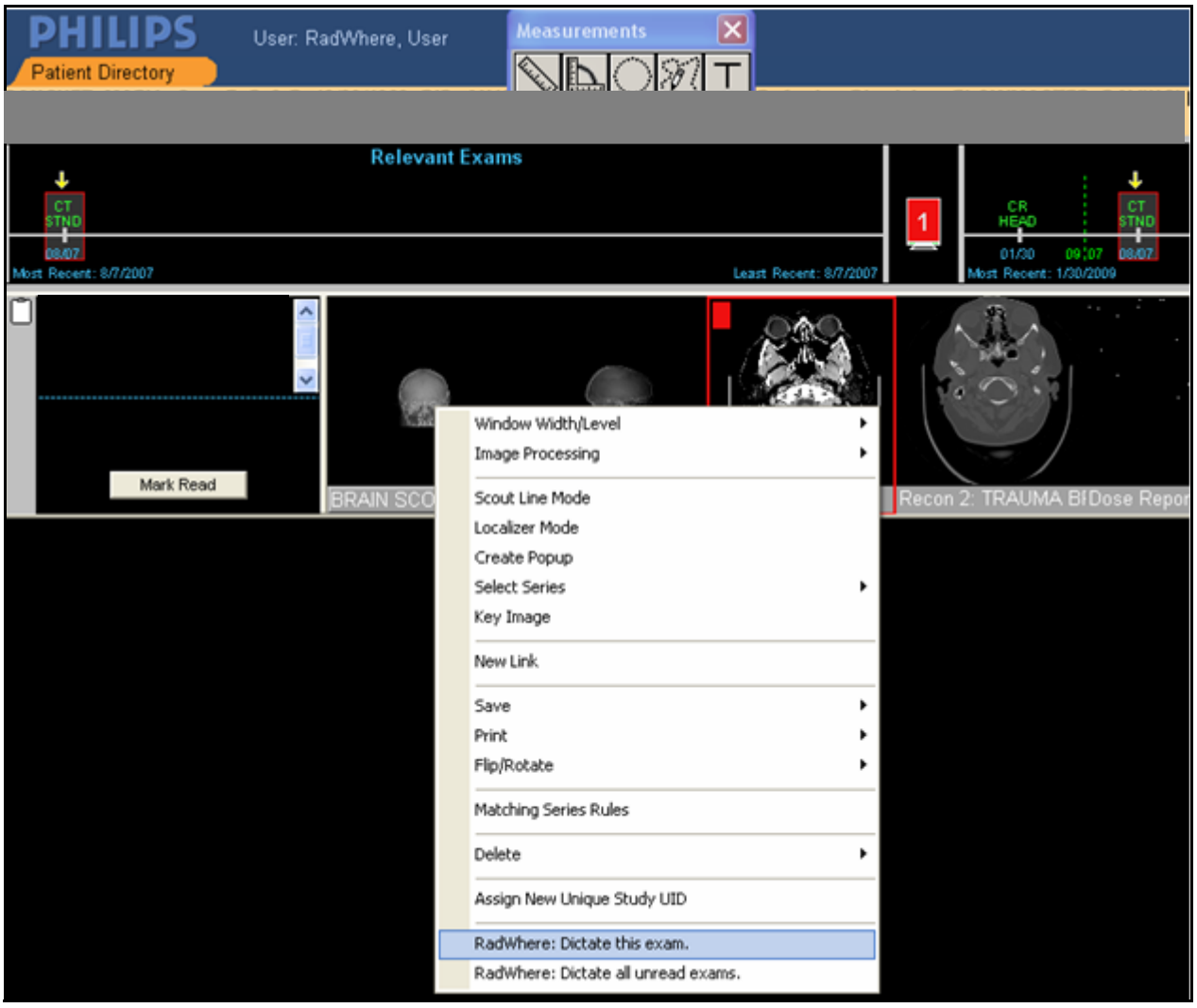
- **Launch from Patient Work List**

  To launch RadWhere from the iSite work list, find a study and right-click on it to bring up the context menu. From here you can click **Dictate this exam** or **Dictate all unread exams**. The latter will search iSite for all unread exams for this patient and load them into a single report (combined result) in RadWhere. (Sensitive Patient information has been blanked out on the following window.)



- **Launch from Image View**

  To launch RadWhere from the iSite image view window, right-click an image thumbnail to bring up the context menu. From here you can click **Dictate this exam** or **Dictate all unread exams**. The latter will search iSite for all unread exams for this patient and load them into a single report (combined result) in RadWhere. (Sensitive Patient information has been blanked out on the following window.)



SDK Guide For Developers

SDK Contents

Item	Description
readme.txt	Documents the contents of the SDK
RadWhereConnectorSetup.msi	Installer for RadWhere integration components: RadWhereCOM.dll, RadWhereAx.dll, and RWCommand.exe.
Samples (Source Code)	**TestRadWhereCOM**: A program to demonstrate the RadWhereCOM API **TestRemote:** Demonstrates the base .Net Remoting API **TestRadWhereAx:** A program to demonstrate the RadWhereAx ActiveX API **RadWherePlugIn**: A web page with embedded RadWhereAx ActiveX control **RadWhereVB**: Visual Basic 6.0 application using RadWhereCOM **XMLGen**: A program demonstrating the use of the DPPlugin DLL for encryption and generating RadWhere and Powerscribe XML files
Documentation	**RadWhereCOM.chm**: Help file for RadWhereCOM API RadWhere Integration Guide.pdf
PACSTester	RadWhere simulator program

Getting Started

1. If not installed already, download and install Microsoft .Net 2.0 SP1.

   This is a prerequisite for the Integration components.

2. Begin by installing the SDK.

   a. Run **SDKSetup.msi,** which will install the files listed above in the SDK Contents section.

   b. Run the **RadWhereConnectorSetup.msi** installer.

   This will install each of the integration components. The COM and ActiveX components will be registered, as well as installed locally.

   RadWhereConnector.msi is installed automatically with each RadWhere Client, but for development purposes, the entire RadWhere installation is not necessary.

3. Install the PACSTester simulator.

   This will enable you to test your integrations without having to install RadWhere.

RWCommand

As part of the Nuance RadWhere application ability to provide integration with PACS and other desktop applications, a simple Windows console application called **RWCommand.exe** is furnished. **RWCommand** is short for RadWhere Command Line. RWCommand makes use of the existing RadWhereURLXML file integration, generating the XML files by using parameters specified on the command line.

Installation

RWCommand.exe is installed to the following directory by default:

\Program Files\Nuance\Nuance RadWhere Integration Component

This location is not included in the **Windows Path**, so your application will need to explicitly define the path, or add this location to the **Windows Path**. The program has no other dependencies (DLLs, assemblies) that are installed with it.

Syntax

Item	Description
-t type	Specifies the type (or name) of the invoking system. If not specified, RWCommand assumes a “Slave mode XML file Integration” (see below); AquariusNET is another type
-f file-path	Complete file path of the XML integration file
-u user-name	Specifies RadWhere logon id
-p password	Password for the user
-a accession	Accession number for RadWhere to open/create a report
-i image-path	File path to image
-s scene-path	File path to scene
-?	Display usage

Slave Mode XML File Integration

One use of RWCommand is to provide **Slave Mode** integration with RadWhere, meaning simple context sharing of the accession(s) opened in PACS. When a PACS User wishes to open the same study in RadWhere, he usually clicks an option in the form of a button, or uses some type of User Interface means, whereupon the PACS can make a command line call to RWCommand. This, in turn, will generate an XML file. Refer to the XML Integration (Slave Mode Only) section on page 34.

You can name the file anything you wish and put it in a directory of your choice. The name and location are configured in RadWhere using the Portal Web Administration application. When RadWhere detects the XML file in the configured directory, it loads the study.

Example:

RWCommand –f d:\Nuance\Order.xml –u joe –a 234567

In this example, an XML integration file would be created in **D:\Nuance**, with a file name of **Order.xml**. The file would contain the logon id and accession number to be launched.

Single Sign-on

If the password parameter is not specified, RadWhere will not attempt to log in the User Name specified. It will simply check to see if the currently-logged on User matches the name specified in the command line. If not, a warning message will display.

If the password parameter is specified, but the password is empty (i.e., nothing after the **–p** option), RadWhere will attempt to log on the specified User without a password. Allowing null passwords to bypass authentication is a site option that must be turned on in the Portal.

When sending a valid password, RWCommand will encrypt the password using the DPPlugin DLL. It will use the encryption key that was configured during installation of the RadWhere Connector Components (page 14). The RadWhereURLXML PACS entry in Portal must also have the same encryption key value specified in the **slave options**.

AquariusNET Integration (Image and Scene Attachments)

When the **AquariusNET** type is specified, RWCommand expects the **–i** and/or **–s** options. These allow the PACS to attach images and/or scenes to the currently opened RadWhere report. Such images can then be launched in the PACS application from RadWhere at any subsequent time.

Example:

RWCommand –i d:\AquariusNet\ct.jpg

In this example, a JPEG image, **ct.jpg**, would be attached to the current report opened in RadWhere.

IRadWhere .Net Remoting Interface

Microsoft .Net remoting is a relatively new way of interprocess communication in the .Net environment. The benefit to this type of integration is that it allows a developer to access the actual object model of the host application, in this case, RadWhere and its IRadWhere interface.

.Net remoting supports the use of TCP/IP, Named Pipes, HTML, or HTTPS as its underlying transport protocol. However, RadWhere has only implemented TCP/IP for transport. This means that your remoting integration must be performed on the same Workstations as RadWhere, or the RadWhere Workstation must have a port open.

IMPORTANT

The remoting port must be set as described in the Set the RadWhere Remoting Port section on page 13 and the RadWhere Connector Components section on page 14. Windows Firewall blocks all TCP/IP ports, so make sure you open the configured port using the Windows Firewall setup utility.

The methods, properties, and events of this object model are documented in the **IRadWhere.chm** help file, located in the RadWhere SDK program files folder.

You may use the **TestRemote** sample (C#) program as a foundation for your .Net remoting application. The key parts to this sample program are documented below.

Establishing a Connection

In order to access the IRadWhere object model, you must connect to the RadWhere listening port using the TCP channel. After establishing the channel connection, instantiate the object by using the **activator** API.

IMPORTANT

*If your application resides on a different Workstation than RadWhere, you will need to modify the URL shown below, replacing **localhost** with the host name of the RadWhere Workstation.*

```
IRadWhere _radWhere;
TcpChannel _channel;

…

//register channel and connect to server
if (null == _channel)
{
    IDictionary props = new ListDictionary();
    props["port"] = 0; //unique listening port (required for callbacks)
    props["name"] = String.Empty; //the Remoting system picks a unique name
    _channel = new TcpChannel(props, null, null);
    ChannelServices.RegisterChannel(_channel, true);
}
_radWhere = (IRadWhere)Activator.GetObject(typeof(IRadWhere),
                "tcp://localhost:9090/Commissure.RadWhere");
```

Registering Events

Once the connection is established, you can register to receive events from RadWhere. The inter-process communication between RadWhere and your application resides on different application threads, so an **event wrapper** is used for each event. These wrappers are defined in the IRadWhere.dll for your convenience.

An example of registering and handling the UserLoggedIn event is shown below.

```
UserLoggedInEventWrapper userLoggedIn_w = new UserLoggedInEventWrapper();
userLoggedIn_w.UserLoggedInArrivedLocally += new
          UserLoggedInHandler(UserLoggedIn);
_radWhere.UserLoggedIn +=new
          UserLoggedInHandler(userLoggedIn_w.LocallyHandleUserLoggedIn);

…

public void UserLoggedIn(object sender, LoginEventArgs e)
{
    MessageBox.Show(e.Username + " is logged in!";
}
```

Making Calls to RadWhere

Once your application is connected and registered for events, you may start making calls to the IRadWhere methods and properties.

The following one-line code snippet demonstrates how to launch a study with accession number **12345** in the current RadWhere site:

_radWhere.OpenReport(_radWhere.Site, "12345"));

RadWhereCOM

If your application is not developed in .Net, or you simply prefer COM programming, you may use **RadWhereCOM.dll** to perform the same functions as the .Net IRadWhere remoting interface. RadWhereCOM is a COM **wrapper** of IRadWhere. When calls are made to RadWhereCOM methods and properties, they are converted into .Net remoting calls directly to RadWhere. For this reason, the listening port must be defined in RadWhere when using this form of integration. Refer to the Set the RadWhere Remoting Port section on page 13.

Documentation of the RadWhereCOM API is located in the **RadWhereCOM.chm** help file located in the RadWhere SDK program files folder.

There are two examples of using RadWhereCOM.

- The first example, **TestRadWhereCOM** is a C# .Net application.
- The second, **RadWhereVB,** is a VB6 (Visual Basic Version 6) application.

The code snippets from RadWhereVB show how to instantiate RadWhereCOM, handle events, and make calls to RadWhere in VB6.

```
Option Explicit

Private WithEvents m_radWhereCOM As RadWhereCOM

Private Sub btnConnect_Click()
   Dim bCanStart As Boolean
   Set m_radWhereCOM = New RadWhereCOM
   m_radWhereCOM.Start bCanStart
End Sub

Private Sub btnCreateNewReport_Click()
   m_radWhereCOM.CreateNewReport txtAccession.text, False
End Sub

Private Sub m_radWhereCOM_ReportFinished(ByVal reportStatus As Long)
   MsgBox "ReportFinished event received; report status = " &
                        CStr(reportStatus)
End Sub
```

Troubleshooting

RadWhereCOM will log error and warning messages in the Windows event log, under the Application group, using the application name **RadWhereCOM**. If you are attempting to troubleshoot problems with RadWhereCOM (e.g., failed connection), and you do not see any messages in the event log, make sure that the event log is not full. You can set up the event log so that it will overlay older messages when it becomes full.

RadWhereAx

RadWhereAx is an ActiveX control **wrapper** of RadWhereCOM. This control was provided mainly for Web applications. In addition to the methods, properties, and events in RadWhereCOM, RadWhereAx also provides a **web listener** to allow **Master Mode** launching of studies in a Web page.

When RadWhere is configured with the **PACS Type RadWhereAx**, both **Master and Slave Modes** are supported.

Since RadWhereAx uses the RadWhereCOM, use the same troubleshooting methods described above (see Troubleshooting on page 28).

Master Mode

When the User clicks the **PACS** button in RadWhere, it means that the User wishes to launch images in the integrated PACS. To achieve this in Web Browser integration is somewhat more complicated than using other forms of integration. To implement Web Browser integration in Master Mode, you must embed RadWhereAx in the Web page. This can be done using the **OBJECT** tag as follows:

<OBJECT id="RadWhereCtrl" height=0 width=0
classid=clsid:c1fa1a92-068c-411f-bf7f-66de29f51a22></OBJECT>

When the Radiologist wishes to launch images in the integrated PACS, he clicks the **PACS** button in RadWhere. RadWhere communicates the accession number(s) of the study by sending an HTML message to RadWhereAx. In order for RadWhereAx to receive this message, you must tell it to **start listening** for HTML messages by calling its **StartListening** method. You must also establish a connection between RadWhereAx and RadWhere by calling the RadWhereAx Start method. These two things are best done early on, such as when loading the page.

```
<body onload="OnLoad()" onunload="OnUnload()" scroll="no" leftmargin=0
      topmargin=0 BOTTOMMARGIN=0 RIGHTMARGIN=0>
…
function OnLoad()
{
      RadWhereCtrl.Start();
      RadWhereCtrl.StartListening();
}
```

When RadWhereAx receives an HTML message from RadWhere, it fires an event called **OpenStudy**. Your Web page must define an event handler for this event so that it can launch the images or perform other processing.

```
function OnOpenStudy(mrn, accessions)
{
	// Load images
	var accessionArray = accessions.split(",");
	for(i=0; i<accessionArray.length; i++)
	{
		// your code goes here
	}
}
…
<SCRIPT FOR="RadWhereCtrl" EVENT="OpenStudy(mrn, accessions)">
   OnOpenStudy(mrn, accessions);
</SCRIPT>
```

In addition to, or in lieu of, using the HTML explicit invocation method explained above, your application can handle other events from RadWhere, including **ReportOpened** to synchronize your application with RadWhere. RadWhereAx supports the full line of events supported by, and documented for, RadWhereCOM.

Slave Mode

Your Web application can share logon and study context with RadWhere in Slave mode by using the same methods and properties used in RadWhereCOM. For example, your Web page may include Client code similar to the following, which is invoked when images are opened in the PACS and the User wishes to dictate the report in RadWhere:

```
function OnViewEvent(user, accessions)
{
	// Log into RadWhere if single sign-on pref is specified
	RadWhereCtrl.Login(user, "");

	// Synchronize accession with RadWhere
	RadWhereCtrl.OpenReport(Accessions);
}
```

The HTML message is sent to RadWhereAx on port **8080**. ***Make sure that this port is not blocked by Windows Firewall or some other security mechanism.***

When calls are made to RadWhereAx methods and properties, they are filtered to RadWhereCOM, which in turn makes .Net remoting calls directly to RadWhere. For this reason, ***the listening port must be defined in RadWhere*** when using this form of integration (see Set the RadWhere Remoting Port on page 13).

RadWhereAx Reference - Methods and Properties

STDMETHOD(Start)(VARIANT_BOOL* pVal);

Launch RadWhere, if necessary, and connect to its remoting interface. This method is synchronous and will not return until RadWhere is ready to accept logins. During the launch, RadWhere is kept minimized.

Returns **true** if RadWhere is successfully connected.

STDMETHOD(Login)(BSTR user, BSTR password);

Log in to RadWhere using the specified Username and Password. RadWhere is made visible and brought to the front in case of User dialogs (e.g., invalid password, microphone wizard, etc.). This method may return before the login process is complete. Use the **LoggedIn** property to determine a successful login.

User - RadWhere user id

Password - User password - if null, bypass authentication

STDMETHOD(CreateNewReport)(BSTR accessionNumber);

Using the specified accession number, creates a new report or opens an existing report with that accession. RadWhere is made visible and brought to the front.

Accession - Accession number

STDMETHOD(get_LoggedIn)(VARIANT_BOOL* pVal);

Returns **true** if someone is logged in to RadWhere.

STDMETHOD(Logout)(void);

Logs the current User out of RadWhere. RadWhere is made visible and brought to the front in case of User dialogs (e.g., Save report, etc.).

STDMETHOD(StartListening)(void);

Starts RadWhereAx HTTP listening port for requests from RadWhere.

STDMETHOD(StopListening)(void);

Stops RadWhereAx HTTP listening port.

STDMETHOD(get_AccessionNumbers)(BSTR* pVal);

Gets the accession numbers that are associated with the open report. It returns **NULL** if no report is open.

STDMETHOD(AssociateOrders)(BSTR accessionNumbers);

Associates the specified accessions with the currently open report.

This request will be ignored if a modal dialog was up at the time of the call.

accessionNumbers - The order filler numbers to associate with the report.

STDMETHOD(CloseReport)(VARIANT_BOOL sign, VARIANT_BOOL preliminary);

Depending on the specified parameter, it either closes the open report without changing its status, or invokes the **Sign** action. When closing, the User is prompted to save, if the report is modified, and is also given the option to cancel the action. When signing, the User may be prompted for confirmation and password.

sign - Indicates whether to just close the report or sign it. Note that **sign**, in this context, depends on user role. It does not always mean that the report will be finalized. For example, when Residents sign, the report enters PendingSignature status.

Preliminary - Indicates whether the report will be signed as preliminary, i.e., its status will be set to PendingSignature instead of Final. This action applies to attendings and this parameter is ignored for Residents who always sign preliminary reports.

STDMETHOD(DissociateOrders)(BSTR accessionNumbers);

Dissociates the specified accessions from the currently open report.

This request will be ignored if a modal dialog was up at the time of the call.

accessionNumbers - The order filler numbers to dissociate from the report.

STDMETHOD(get_Minimized)(VARIANT_BOOL* pVal);

Gets the Window minimized state.

STDMETHOD(put_Minimized)(VARIANT_BOOL newVal);

Sets the Window minimized state.

STDMETHOD(OpenReport)(BSTR siteName, BSTR accessionNumbers);

Creates or opens a report for the specified accessions. To verify success, wait for the UserLoggedIn event.

siteName - The name of the site that the accession numbers belongs to. Specify blank for system-wide search, inclusing all the sites that the user has access to. Specify **NULL** to search according to the current site selection in Explorer screen.

accessionNumbers - Comma-delimeted string containing the order filler numbers to associate with the report.

STDMETHOD(PreviewOrders)(BSTR siteName, BSTR accessionNumbers);

Searches for and previews the specified orders. If multiple orders are found, only the first is previewed. If the order is reported, the report is also previewed.

siteName - The name of the site that the accession numbers belongs to. Specify blank for system-wide search, inclusing all the sites that the user has access to. Specify NULL to search according to the current site selection in Explorer screen.

accessionNumbers - The order filler numbers.

STDMETHOD(Terminate)(void);

Terminates the application.

This call will prompt the user to save/close an open report and will prompt the user to terminate. If the user cancels this action, the application will not terminate.

STDMETHOD(get_Username)(BSTR* pVal);

Gets the username of the logged-on user. It returns **NULL** if no one is logged on.

STDMETHOD(get_SiteName)(BSTR* pVal);

Gets the name of the site that the open report belongs to. It returns **NULL** if no report is open.

STDMETHOD(get_AlwaysOnTop)(VARIANT_BOOL* pVal);

Gets whether the application should is displayed as the topmost window.

STDMETHOD(put_AlwaysOnTop)(VARIANT_BOOL newVal);

Sets whether the application should be displayed as the topmost window.

STDMETHOD(Stop)(void);

Terminates the application.

This call will prompt the user to save/close an open report and will prompt the user to terminate. If the user cancels this action, the application will not terminate.

RadWhereAx Reference - Events

HRESULT __stdcall ReportFinished(int reportStatus)

This event is fired when a User closes a report in RadWhere. It can occur by saving, signing or discarding a report.

HRESULT __stdcall ReportOpened(BSTR SiteName, BSTR AccessionNumbers)

Fires when the report is opened for editing.

HRESULT __stdcall ReportClosed(BSTR SiteName, BSTR AccessionNumbers)

Fires when the report is closed (i.e., saved or discarded).

HRESULT __stdcall AccessionNumbersChanged(BSTR SiteName, BSTR)

Fires when the report is associated with new orders.

HRESULT __stdcall UserLoggedIn(BSTR Username)

Fires when the User logs in.

HRESULT __stdcall UserLoggedOut(BSTR Username)

Fires when the User logs out.

HRESULT __stdcall Terminated()

Fires when the application is closed.

RadWhere URL/XML Integration

URL Integration (Master Mode Only)

RadWhere can be configured with a custom URL to support Slave Mode integration. RadWhere can launch images from a PACS in a Web browser by specifying the accession number and/or patient id, and it can also include the current login id and password of the logged in RadWhere user.

For example, PACS supports a launching URL in the following format:

http://OurPACS.com/showImage?accession=12345&u=user&password=pw

RadWhere would be configured with the following master argument:

http://OurPACS.com/showImage?accession={accession}&u={user}&password={password}

XML Integration (Slave Mode Only)

The simplest and easy way to achieve context sharing of studies (i.e., accessions) is to use the XML **File Drop** approach. Using this form of integration, the PACS can support single sign-on, using login ids and encrypted passwords, as well as launch studies by one or more accession numbers.

Encryption is performed using the **DPPlugin** encryption DLL. Refer to the Encryption section on page 35 for details.

Example:

```
<DictationData>
    <New>
        <UserID>joe</UserID>
        <Password>tABvALgAcwDnAFAAUQBdAH4BEwBHAA==</Password>
        <AccessionNumber>123456,435423</AccessionNumber>
    </New>
</DictationData>
```

Single Sign-on Notes

- If the Password element is not specified, ***RadWhere will not attempt single sign-on***. Instead, it will validate the login ID specified and display a warning message to the User if the specified login ID does not match the currently logged in RadWhere User.
- If the Password element **is** specified, but it is **empty or blank**, this tells RadWhere to log the User on without password verification. Password verification bypass must be configured in the Portal in order for this to work.
- If a password is specified, it must be encrypted using the DPPlugin EncryptData method, and **encoded with Base64 encoding**.
- If using an encrypted password, and the XML file is created by RWCommand (see the RWCommand section on page 23), RWCommand will perform the encryption using the RWCommand encryption key configured at installation time (see the RadWhere Connector Components section on page 14).

Powerscribe XML Integration

RadWhere supports a Powerscribe-compatible, two-way XML file drop interface. Details of this schema and use of encryption are described below.

Take note of the configuration options documented earlier in this documentation. The XML schema shown below may be customized on site for certain options, such as the tag name of the accession number element and the multiple accession separator character.

Encryption

Single sign-on is supported using encrypted field(s). To encrypt the logon id and password, the encryption DLL will be installed and registered on the RadWhere workstation as part of the initial installation process. If you are using the SDK, it will be installed and registered during installation of the RadWhere Components.

The encryption DLL is a COM component providing the following two methods:

1. BSTR EncryptData(BSTR data, BSTR encryptionPassword)
2. BSTR DecryptData(BSTR data, BSTR encryptionPassword)

Both the PACS and RadWhere must use the same encryption key for this to work. The encryption key is entered as part of the slave options in the RadWhere Portal.

Slave Mode

This integration supports context sharing of accession number(s), user name and password.

Input (Request) File Format

To log in a user on RadWhere, create an XML file similar to the one below.

```
 <?xml version="1.0" ?>
<ChunkData>
    <Event>
      <Event1>Login</Event1> <!-- Either "login" or "logout" -->
      <User>
       <LoginID>tABvALgAcwDnAFAAUQBdAP0A7wAwAA==</LoginID>
       <Password>tABvALgAcwDnAFAAUQBdAP0A7wAwAA==</Password>
       </User>
    </Event>
</ChunkData>
```

Note that both the logon id and password are encrypted using the DPPlugin and encoded in base64.

Below is an example of a file that an application would drop in the RadWhere folder to launch a study for accessions 10566 and 9988785.

```
<?xml version="1.0" ?>
<ChunkData>
  <Fields>
    <Field6>10566,9988785</Field6>
  </Fields>
</ChunkData>
```

Output (Status) File Format

When a report has been closed or saved in RadWhere, a **status** file is dropped in a shared folder to indicate the status change for the given accession(s). Below is an example of an XML file that RadWhere would drop in your application folder to indicate that the user has signed a report for studies with accession numbers 2435 and 2346:

```
<?xml version="1.0" ?>
<ChunkData>
 <AccessionNumber>2435,2436</AccessionNumber>
 <Event>
   <Event1>signed</Event1><!--Either "dictated","signed","opened","deleted" -->
 </Event>
</ChunkData>
```

Master Mode

Below is an example of an XML file that RadWhere would drop in your application folder to indicate that the user has opened a report for studies with accession numbers 2435 and 2346:

```
<?xml version="1.0" ?>
<ChunkData>
 <AccessionNumber>2435,2436</AccessionNumber>
 <Event>
   <Eventl>opened</Eventl><!--Either "dictated","signed","opened","deleted" -->
 </Event>
</ChunkData>
```

Document Revision History

Note: In this table the most recent changes are first by date.

Date	Page	Change (Paragraph, Sentence, Figure, Table, etc.)	Initials
04/08/10	All	First Formal Publication	AFW