iSite Radiology ActiveX Reference Guide

iSite PACS 3.6 and 4.1

Rev. 1.10
2009 Oct 15

Philips Healthcare Informatics, Inc.
Enterprise Imaging

PHILIPS

PHILIPS

Enterprise Imaging

iSite Radiology ActiveX Reference Guide

iSite PACS 3.6 and 4.1

Rev. 1.10

2009 Oct 15

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

Table of Contents

1 Introduction

The iSite Radiology application is an executable file that contains an ActiveX control that is the iSite Radiology program. The ActiveX control can be used in other applications with limitations.

This document is a comprehensive list of all properties, methods, and events available to the developer through the iSite PACS Radiology API. For general information on using the iSite PACS Radiology API, including overviews, implementation methods and design considerations, reference "*iSite PACS Client and API Overview.doc"* in the Philips iSite PACS SDK.

All properties, methods, and events in this document are supported for iSite PACS Client versions 4.0 and higher unless otherwise specified in the method description.

Note: This document applies to iSite PACS 4.1.23 and higher and iSite PACS 3.6.

2 iSiteRadiology Control

The **iSiteRadiology** control provides access to the iSite PACS Server application and control over the iSite Radiology application. Only one instance of iSite Radiology is allowed to run at a time on a computer.

Note: The **iSiteRadiology** control cannot be resized with the wrapping application.The API functions ShowDebugWindow(), ShowiQueryWindow() and ShowPreferenceDialog() are not supported if the ActiveX control is contained in the other application.

2.1 Properties

iSiteRadiology contains properties that are configurable via **iSite.ini** file located in **iSiteRadiology.exe** module directory.

Name	Type	Default	Description
iSyntaxServerIP	BSTR	Empty String	IP address of the main iSyntax Server.
iSyntaxServerPort	BSTR	"6464"	Port number to use to communicate with the server.
ImageSuiteURL	BSTR	Empty String	URL of the Imaging Suite server
ImageSuiteDSN	BSTR	Empty String	Imaging Suite Data Source Name
FailoveriSyntaxServerIP	BSTR	Empty String	Fail over address of the iSyntax Server
StartupMessageURL	BSTR	Empty String	URL of a custom start up message to be displayed when the user login in

Name	Type	Default	Description
BrowserPageName	BSTR	Empty String	Custom BrowserPage Tab that is displayed instead of the Patient Directory Tab, Shortcuts and Folder List. BrowserPageURL must also be set and the HideFolder option must also be set.
BrowserPageURL	BSTR	Empty String	Full path of the URL of the custom BrowserPage to be displayed.
CacheSizeMB	long	0	Specifies the amount of memory to reserve for caching images. Set to 0 for the default. The default is half the amount of physical RAM in the system or 32mb whichever is greater.
WorkstationLocation	BSTR	“”	Location of this workstation. This may be different in sites that have remote access or WAN connectivity. Use **GetWorkstationLocations()** to retrieve a list of valid locations.
*** SecurityCodes**	BSTR	Empty String	Deprecated. SecurityCodes is set to “”
*** Initialized**	BOOL	False	Is set to **True** when the control is successfully loaded and connects to the server.
Options	BSTR	Empty String	See below for details
CCOWOptions	BSTR	Empty String	Sets the values for the Context or Patient and User values.
CCOWEnabled	BOOL	False	Set to True to enable the **Clinical Context Object Workgroup** protocol as defined by the HL7 standards

* - Read only property

Sample

3.6 mode (communicating with IDX Backend)

```
[Server]
iSyntaxServerIP = “192.168.55.1”
iSyntaxServerPort = “6464”
ImageSuiteURL = “http://192.168.55.1/iSuite”
ImageSuiteDSN = “iSite”
Options = “”
```

4.1 mode (communicating with Philips Backend)

```
[Server]
iSyntaxServerIP = “192.168.55.1”
iSyntaxServerPort = “6464”
```

```
ImageSuiteURL = “http://192.168.55.1/iSiteWeb/WorkList/PrimaryWorkList.ashx”
ImageSuiteDSN = “iSite”
Options = “StentorBackEnd”
```

Options Parameters

The options property provides a simple mechanism to enable or disable various features or functions of the control. Using commas specifies multiple options. The following options are supported:

Parameter	Description
StentorBackEnd	Starting with iSite PACS 3.6/4.1, this option is used to initialize the client in the mode to communicate with the Philips Backend. Without this option, the client will default to IDX Backend communication mode.
DisableAutoLogout	Prevents the system from automatically logging the user out.
EnableFireExamMarkedReadEvent	This option has to be deprecated.
IDXMode or IDXModeX	Runs iSite Radiology in IDX mode = IDX logo will be displayed.
HideFullUserName	Hides the currently logged in user name.
ForceRawDICOMPS	Forces the application to always load the Raw DICOM Presentation state instead of a saved Presentation State.
HideCloseTabButton	Hides the Close button on the Canvas Page.
HideFolder	Hides the Patient Directory Folder tab.
HideReportButton	Hides the Report button.
DisableExamHistory	Hides the History folder.
DisableTipOfTheDay	Hides the “Tip of the Day.”
HideLogoutButton	Hides the Logout button.
ForcePatientMode	Forces Patient Mode and disables the preference to change to exam mode.
NoMessageBox	No message boxes displayed.
ShowMonitors	Set this option if you want all images to be displayed on secondary monitors exclusively.

Note: The **ShowMonitors** option has some implications. If the **ShowMonitors** option is set and the **OpenCanvasPage** API method is called with the reveal parameter set to false, there can be a maximum of one open Canvas Page at a time. The presence of the **ShowMonitors** option also implies the usage of the **BrowserPageName** and **BrowserPageURL** properties. In this case it is assumed that your plug-in will be overriding the default Worklist page.

Sample

```
Options = “StentorBackEnd,HideFolder,DisableAutoLogout”
```

Go to Table of Contents

2.2 Methods

2.2.1 Initialize

This method is used to initialize iSite Radiology programmatically.

```
boolean Initialize();
```

Parameters

None.

Return Values

Value	Meaning
True	Success
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

3,4,91

Go to Error Codes

Remarks

This method must be invoked successfully before any other method can be invoked. Any changes to the control properties are ignored after this method has been invoked successfully.

Go to Table of Contents

2.2.2 Login

This method logs into the server using the specified name and password.

```
boolean Login(
     BSTR UserName,
     BSTR Password,
     BSTR AuthSource,
     BSTR Token,
     BSTR Mnemonic);
```

Parameters

Name	Description
UserName	The username.
Password	The password.
AuthSource	Contains either iSite PACS (same authentication as before) or IDXNTLM for NT domain authentication.
Token	Empty String.
Mnemonic	A name uniquely identifying this user account. May be an empty string. This name is used for logging purposes only.

Return Values

Value	Meaning
True	Success, user logged in
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

0,5,6,7,8,9,17

Go to Error Codes

Remarks

If a user is currently logged in, you must logout before calling this function. The **Mnemonic** may be used to identify a user while logging into iSite PACS using a single user name and password.

Go to Table of Contents

2.2.3 Logout

This method logs the current user out.

```
boolean Logout();
```

Parameters

None.

Return Values

Value	Meaning
True	Success, user logged out.
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

This function does not have an error code.

Go to Error Codes

Remarks

Calling logout when no user is logged in results in a success.

Go to Table of Contents

2.2.4 ChangePassword

This method will change the password of the currently logged in user.

```
boolean ChangePassword(
      BSTR OldPassword,
      BSTR NewPassword);
```

Parameters

Name	**Description**
OldPassword	The old password.
NewPassword	The new password.

Return Values

Value	**Meaning**
True	Success, password was successfully changed
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

0, 10,17, 200, 201, 202

Go to Error Codes

Remarks

You must first login before calling this function.

Go to Table of Contents

2.2.5 Query

This function runs a query against the patient exam database and returns the results. To return a sorted query result list, use the **QueryEx()** function.

```
BSTR Query(
     BSTR Query,
     BSTR Type,
     LONG MaxResults);
```

Parameters

Name	Description
Query	Query string
Type	One of:

Name	Description
INTERPRETATION	This type of query is run against the most recently acquired exams (last 30-90 days). This type of query is faster than a LOOKUP type, but will not return exams older than 90 days. User must have ISTANYPAT permission or this will fail.
LOOKUP	This type of query is run against the entire exam database. Since the database may contain millions of exams, it is slower than an INTEPRETATION and should be avoided unless necessary. In some cases, it is absolutely necessary such as getting a patient's exam history or a query for a really old prior. User must have ISTANYPAT permission or this will fail
REFERRING	This type of query is just like a LOOKUP query, but only those exams that are assigned to the current user are returned. Use this if the user does not have ISTANYPAT permission.
EXCEPTION	This type of query is run against the exception worklist. If a DICOM study does not resolve to an Exam, an entry is added to the exception worklist until it is resolved. Since exceptions should be rare, you should avoid this type of query unless you need to know about unresolved studies. User must have ISTEXCEPT security.

Name	Description
MaxResults	The maximum number of exams the query should return. Valid range is 1-1000. WARNING = Returning a large number of results can result in serious performance degradation, please check with Philips Global PACS ActiveX support if you need to query for more than 200 exams. If the method returns MaxResults+1 results, this indicates there are more results on the server that meet the search criteria, returning MaxResults or less results indicates that all records that matching the criteria have been returned.

Return Values

Value	Meaning
XML	Results of the query
“”	Empty string = error occurred, check **GetLastErrorCode()**

Error Codes

0,6,10,11,12,13,17

Go to Error Codes

Remarks

The Query method is used to find exams in the system that matches a certain criteria. A match occurs when all exam attributes specified in the query exactly match. The results of the query will contain all of the attributes for that query type. Note that the **IDXIntExamID** is interchangeable with the **IntExamID**.

Note the following:

- The attributes for the Query event are different for iSite PACS 3.6 and 4.1. In iSite PACS.4.1, when a LOOKUP, INTERPRETATION, or REFERRING query is executed, a range from 30 days ago to that day is used when retrieving the date except for the following filters:
  - x00100020 (MRN)
  - x00080050 (Accession #)
  - x00100030 (Patient’s date of birth)
  - LockedByName (only All Locked Exams)

  To retrieve data from the past, use the BETWEEN clause and enter a start and end date. These dates must also be no more than 30 days apart
- The EXCEPTION query has the same behavior in iSite PACS 3.6 and 4.1.
- The INTERPRETATION query in iSite PACS 4.1 has the same behavior as LOOKUP, meaning that it looks into a 30 day range back from the current date.

Attributes for LOOKUP, REFERRING and INTERPRETATION Queries

XML tag	Description	Data Format	Comment
x00100010	Patient Name	String	
x00100020	Patient ID (MRN)	String	Not restricted to a 30 day range in iSite PACS 4.1
x00100030	Patient’s Birth Date	YYYYMMDD	Not restricted to a 30 day range in iSite PACS 4.1
x00100040	Patient’s Sex	M, F, U	Not supported in iSite PACS 4.1
StudyDTTM	Exam Date and Time	YYYY-MM-DD HH:MM:SS	

XML tag	Description	Data Format	Comment
x00080050	Accession Number	String	Not restricted to a 30 day range in iSite PACS 4.1
x00080090	Referring Physician's Name (for iSite PACS 3.6 only)	String	Not supported in iSite PACS 4.1
x00180015	Body Part Examined	String	
x00080060	Modality	String	
x00081032_1	Procedure Code	String	
x00081032_2	Procedure Description	String	Not supported in iSite PACS 4.1
x00081080	Admitting Diagnosis Description		Not supported in iSite PACS 4.1
IsStatExamFLAG	Stat. "Y" if exam is a "STAT" exam, "N" otherwise.	String. Either "Y" or "N"	
IDXExamStatus	Exam Status. One of the following:	String	
	Value	Meaning	
	O	Ordered	
	S	Scheduled	
	I	In Progress	
	C	Completed	
	D	Dictated	
	P	Preliminary	
	F	Finalized	
	A	Addended	
	R	Revised	
	!	Exception	
	X	Cancelled	
	NULL	Deleted	
	N	Non-reportable	

XML tag	Description	Data Format	Comment
LockedByName	LockedByName=" Name" or	String	Not restricted to a 30 day range in iSite PACS 4.1 for All Locked Exams
	Value		
	>""	All Locked Exams	
	=""	All Not Locked	
PatientLocation	Patient location	String	
HasImagesFLAG	"Y" if exam has images available to be launched in iSite, "N" otherwise.	String. Either "Y" or "N". Not Valid for EXCEPTION queries.	
IDXIntReferringPhysID	IDX unique Referring Physician ID	Number	Not supported in iSite PACS 4.1
IDXIntPatientID	IDX unique Patient ID	Number	
IDXIntExamID	IDX unique Exam ID	Number	
OrganizationCode	Organization Code	String	
PerformingResource	The name of the equipment that was used to perform the exam	String, 10 characters	
SubspecialtyCode	Query SubspecialtyCode , Parse XML for SubspecialityCode	String, 10 characters	
ExamReadFLAG	"Y" if exam was read, "N" otherwise.	String. Either "Y" or "N"	Not supported in iSite PACS 4.1

Attributes for EXCEPTION Queries (same in iSite PACS 3.6 and 4.1)

XML tag	Description	Data Format
x00100010	Patient Name	String
x00100020	Patient ID (MRN)	String
x00100030	Patient's Birth Date	YYYYMMDD
x00100040	Patient's Sex	M, F, U

XML tag	Description	Data Format
StudyDTTM	Exam Date and Time	YYYY-MM-DD HH:MM:SS
x00080050	Accession Number	String
x00081030	Study Description	String
x00180015	Body Part Examined (Series)	String
x00080060	Modality (Series)	String
x00081032_1	Procedure Code	String
X00081032_2	Procedure Description	String
IDXIntExceptionID	IDX unique Exception ID	Number

Note: in iSite PACS 4.1, when the binary operators (<, >, <=, or >=) are used with a date, a 30 day range is considered. For example, x00100030 > 19000101 would check between January 1, 1900 and January 30, 1900, not the entire database after January 1, 1900.

A query string consists of a series of attribute/operator/value pairs separated by the word AND. Values should be enclosed in double quotes. The available operators are:

=	**Equals (exact match)**
LIKE	Partial match. In iSite PACS 4.1, not supported with Accession # (x00080050). Treated as an equal sign.
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
BETWEEN v1 AND v2	Range match

Additional operation available for iSite PACS 3.6

IN(V**1,V2,V3…)**	Multiple Value Match.

Examples

Find all exams for the patient with MRN “12345”:

```
x00100020 = “12345”
```

Find all exams for patients with last names beginning with “Smi”:

```
x00100010 LIKE “Smi”
```

Find all CT exams in the last three hours (assuming current date time is Apr 20, 2001 5:42 PM):

```
x00080060 = “CT” AND x00080020 > “20010420” AND x00080030 BETWEEN “134200” AND
“174200”
```

Find all stat exams in the ER:

```
IsStatExamFLAG = “Y” AND PatientLocation=”ER”
```

Find all completed MR exams:

```
IDXExamStatus = “C” AND x00080060 = “MR”
```

Go to Table of Contents

2.2.6 QueryEx

This function runs a query against the patient exam database and returns the results sorted according to the input parameters.

```
BSTR QueryEx(
        BSTR Query,
        BSTR Type,
        BSTR PrimarySort,
        LONG PrimarySortDir,
        BSTR SecondarySort,
        LONG SecondarySortDir,
        LONG MaxResults);
```

Parameters

Name	Description
Query	The Query string. See Query for details.
Type	The Query type. See Query for details.
PrimarySort	The primary sort key.
PrimarySortDir	0 – Ascending order; 1 – Descending order
SecondarySort	The secondary sort key.
SecondarySortDir	0 – Ascending order; 1 – Descending order
MaxResults	The maximum number of exams the query should return. Valid range is 1-1000. WARNING – Returning a large number of results can result in serious performance degradation, please check with Philips Global PACS ActiveX support if you need to query for more than 200 exams.

Return Values

Name	Meaning
XML	Results of the query
“”	Empty string – error occurred, check GetLastErrorCode()

Error Codes

0,6,10,11,12,13

Go to Error Codes

Remarks

The QueryEx method is used to find exams in the system that matches a certain criteria and return the exams sorted according to the input parameters. A match occurs when all exam attributes specified in the query exactly match. See Query function remarks for details on the query attributes. The tables below specify the attributes for the sorting parameters. The results of the query will contain all of the attributes for that query type sorted according to the input parameters.

Note

- IDXIntExamID is interchangeable with the IntExamID.
- IDXIntExceptionID is interchangeable with the IntExceptionID.

Valid sort parameters for LOOKUP, REFERRING and INTERPRETATION queries:

XML tag	**Description**	**Data Format**	
x00100010	Patient Name	String	
x00100020	Patient ID (MRN)	String	
x00100030	Patient’s Birth Date	YYYYMMDD	
x00100040	Patient’s Sex	M, F, U	
StudyDTTM	Exam Date and Time	YYYY-MM-DD HH:MM:SS	
x00080050	Accession Number	String	
x00180015	Body Part Examined	String	
x00080060	Modality	String	
x00081032_1	Procedure Code	String	
x00081032_2	Procedure Description	String	
x00081080	Admitting Diagnosis Description		
IsStatExamFLAG	Stat. “Y” if exam is a “STAT” exam, “N” otherwise.	String. Either “Y” or “N”	
IDXExamStatus	Exam Status.	String. One of the following.	
		Value	Meaning
		O	Ordered
		S	Scheduled
		I	In Progress
		C	Completed
		D	Dictated
		P	Preliminary
		F	Finalized
		A	Addended
		R	Revised
		!	Exception

XML tag	Description	Data Format	
		X	Cancelled
		NULL	Deleted
		N	Non-reportable
LockedByName	LockedByName="Name"	String	
PatientLocation	Patient location	String	
HasImagesFLAG	"Y" if exam has images available to be launched in iSite, "N" otherwise.	String. Either "Y" or "N". Not valid for EXCEPTION queries.	
IDXIntReferringPhysID	IDX unique Referring Physician ID.	number	
IDXIntPatientID	IDX unique patient ID	number	
IDXIntExamID	IDX unique exam ID	number	
OrganizationCode	Organization Code	String	
SubspecialtyCode	Query SubspecialtyCode	String, 10 characters	
PerformingResource	The name of the equipment that was used to perform the exam	String, 10 characters	
ExamReadFLAG	"Y" if exam is read, "N" otherwise	String. Either "Y" or "N"	

Valid Sort Parameters for EXCEPTION Queries

XML tag	Description	Data Format
x00100010	Patient Name	String
x00100020	Patient ID (MRN)	String
x00100030	Patient's Birth Date	YYYYMMDD
x00100040	Patient's Sex	M, F, U
x0020000D	Study Instance UID	DICOM UID
StudyDTTM	Study Date and Time	YYYY-MM-DD HH:MM:SS
x00080050	Accession Number	String
x00081030	Study Description	String
x00180015	Body Part Examined (Series)	String
x00080060	Modality (Series)	String
x00081032_1	Procedure Code	String
PerformingResource	The name of the equipment that was used to perform the exam	String, 10 characters
X00081032_2	Procedure Description	String

XML tag	Description	Data Format
IDXIntExceptionID	IDX unique exception ID	number

Go to Table of Contents

2.2.7 Exists

This function queries the exam database and returns the number of matching exams.

```
long Exists(
      BSTR Query,
      BSTR QueryType);
```

Parameters

Name	Description
Query	The query string, See Query()
QueryType	One of: **Name** — **Description** LOOKUP — Patient lookup query. User must have ISTANYPAT permission or this will fail. REFERRING — Referring worklist query, only exams where this user is the referring physician are returned. EXCEPTION — Queries the exception worklist. User must have ISTEXCEPT security.

Return Values

Name	Meaning
Positive #	The number of exams matching the query
0	No matching exams, check for error with **GetLastErrorCode()**

Error Codes

0,6,10,11,12,13,17

Go to Error Codes

Remarks

This function can be used to quickly determine if the iSite PACS database contains specified exams.

Go to Table of Contents

2.2.8 FindPatient

This function queries the database for the internal Patient ID for a given a MRN and Organization.

```
BSTR FindPatient(
      BSTR MRN,
      BSTR Organization);
```

Parameters

Name	Description
MRN	MRN
Organization	The Organization in which this MRN is valid.

Return Values

Name	Meaning
String	The internal Patient ID
""	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,6,10,12,17

Go to Error Codes

Remarks

This function converts an external MRN or Patient ID to an internal Patient ID.

Go to Table of Contents

2.2.9 FindExam

This function queries the database for the internal Exam ID given an Accession Number, MRN and Organization.

```
BSTR FindExam(
        BSTR Accession,
        BSTR MRN,
        BSTR Organization);
```

Parameters

Name	Description
Accession	The Accession Number for this exam.
MRN	The MRN for this Accession.
Organization	The Organization in which this MRN is valid.

Return Values

Name	Meaning
String	The internal Exam ID
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,6,10,12,17

Go to Error Codes

Remarks

Some RIS/HIS systems reuse accession numbers, which prevents them from being a unique key.

Go to Table of Contents

2.2.10 FindStudy

This function queries the database for the internal Exam ID given a DICOM Study Instance UID.

```
BSTR FindStudy(BSTR StudyUID);
```

Parameters

Name	Description
StudyUID	The DICOM Study Instance UID.

Return Values

Name	Meaning
String	The internal Exam ID
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,6,10,17

Go to Error Codes

Remarks

In some cases, a **StudyUID** may map to more than one Exam. In this case, this method will return one of the matching **InternalExamID’s**. This method does not work for exceptions. If no matching exams are found, an empty string is returned and the error code is set to 0 (no errors).

Go to Table of Contents

2.2.11 GetReportData

This function returns the diagnostic report for the specified exam.

```
BSTR GetReportData(
     BSTR IntPatientID,
     BSTR IntExamID);
```

Parameters

Name	Description
IntPatientID	The internal Patient ID. (Legacy parameter. Currently ignored)
IntExamID	The internal Exam ID.

Return Values

Value	Meaning
Text	XML string that contains raw report text
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,6,10,17,100,101

Go to Error Codes

Remarks

The **IntPatientID** parameter is a legacy parameter and is currently ignored.

The format of the report data depends upon the RIS system. It may contain embedded formatting information such as HTML. It is possible that exam specified by **IntExamID** has multiple reports. Returned XML string has next format:

```
<AllReports>
     <Report>
            … RAW REPORT DATA
     </Report>
     <Report>
            …RAW REPORT DATA
     </Report>
    .
.
</AllReports>
```

See also

FindPatient, FindExam

Go to Table of Contents

2.2.12 SetPreference

This method is used to store a User, System, or Machine preference

```
boolean SetPreference(
      BSTR Name,
      BSTR Type,
      BSTR Data);
```

Parameters

Name	Description
Name	The unique name for this preference.
Type	One of:

Name	Description
User	Stores this as a User preference
System	Stores this as a System preference
Machine	Stores this as a Machine preference

Data	Data for this preference, (an XML string).

Return Values

Value	Meaning
True	Success
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

0,1,3,4,10,17

Go to Error Codes

Remarks

The name of the preference should be as unique as possible to avoid collisions with other plug-ins or integrators. Passing a null string for Data will remove the preference.

Go to Table of Contents

2.2.13 GetPreference

This method is used to retrieve a stored User, System, or Machine preference

```
BSTR GetPreference(
     BSTR Name,
     BSTR Type);
```

Parameters

Name	Description
Name	The unique name for this preference.
Type	One of:

Name	Description
User	Stores this as a User preference
System	Stores this as a System preference
Machine	Stores this as a Machine preference

Return Values

Value	Meaning
BSTR	XML String containing the preference(s). The string is empty if an error occurred or there is no preference.

Error Codes

0,1,3,4,10,17,18

Go to Error Codes

Remarks

The name of the preference should be as unique as possible to avoid collisions with other plug-ins or integrators.

Go to Table of Contents

2.2.14 GetLastErrorCode

Returns the error code for the last error that occurred, or 0 if the last API invocation succeeded.

```
long GetLastErrorCode();
```

Parameters

None.

Return Values

Value	Meaning
0	No Errors
1	Non Visual Control Already Initialized
2	Non Visual Control Not Initialized
3	User Logged Out
4	License Expired
5	Error, user already logged in
6	Communication Error with Imaging Suite
7	Communication Error with iSyntaxServer
8	Failure, invalid name or password
9	Failure, user does not have permission
10	Failure, user not logged in
11	Invalid Query String
12	Failure, cannot query because user has ISTNOREP security.
13	Invalid Query Type
14	Invalid Imaging Suite DSN
15	Invalid DICOM Tag
16	Radiology control not initialized
17	Invalid or missing parameter
18	Preference not found
19	Failed to create Media Export Page
20	Media export already visible
90	Invalid Session Object
91	Already Initialized

Value	Meaning
100	Invalid IntPatient ID
101	Invalid IntExamID
102	Clinical Exam notes enabled
103	Clinical Exam notes disabled
104	Invalid Exception Key
105	Invalid Study UID
106	Invalid Series UID
110	Unknown menu item
111	Menu item already exists
120	Unknown Window ID
121	Invalid Image UID
122	Invalid Zoom Level
123	Invalid Coordinate
124	Page is not visible
125	Page doesn't exist
126	Shelf doesn't exist
127	Exam is not locked
128	Exam is locked already
129	Exam was not locked by current user
130	Exam is marked read already
131	Exam is not ready for dictation (Must have a status InProgress or Completed)
132	Main exam closed
133	Page already exists
200	Invalid old password
201	Invalid new password
202	New Password not different from old password
203	New password has invalid length
204	No folder tree present.
205	Media Export folder not available to the current logged in user.
206	No exam available in the media export folder for export.
207	Media Export Error when failed to move the files to the Media Export Folder

Value	Meaning
208	Media Export Error while downloading the zip file
209	Media Export Error while writing the exams to the CD Manager
210	Media Export Error while downloading the exam(s) to local directory (when user doesn't have the write permission
211	Media Export Error communication with the Server
212	Media Export Error communicating with the IDX Server
213	Media Export Error while reading the Study File
214	Media Export Error while writing the Study File to the file system
215	Media Export error opening the Image
216	Function not allowed in FailOver mode
217	The file could not be opened
218	The file could not be written
219	Invalid location
220	Unknown Annotation ID
221	iExport is not running
222	iQuery is not running
223	The server failed to change the password
224	Out of Memory
225	Language Not Supported
227	Compression Enabled
999	Unsupported
1000	Internal - unexpected

Remarks

Invoke this method to determine the error code for the method call that failed. The error code is reset to 0 (No Errors) if an API invocation succeeds.

Go to Table of Contents

2.2.15 Reset

This method closes all open Patient tabs and returns the control to its initial state.

```
boolean Reset();
```

Parameters

None.

Return Values

Value	Meaning
True	Success
False	Failure, check error with **GetLastErrorCode()**

Error Codes

0,10

Go to Error Codes

Remarks

None.

Go to Table of Contents

2.2.16 ListCanvasPages

This method returns a list of the **CanvasPageID**’s for the open Canvas Pages.

```
BSTR ListCanvasPages();
```

Parameters

None.

Return Values

Value	Meaning	
BSTR	XML encoded string with the Canvas Page IDs	
	CanvasPageIDs	List of Canvas Page IDs
	CanvasPageID	Canvas Page ID

Error Codes

0,1,3,4,10, 125

Go to Error Codes

Remarks

None.

Go to Table of Contents

2.2.17 OpenCanvasPage

This function opens the specified exam in a new Canvas Page.

```
BSTR OpenCanvasPage(
     BSTR IntExamID,
     BSTR IntExceptionID,
     boolean Reveal,
     boolean Lock,
     boolean OpenNew);
```

Parameters

Name	Description
IntExamID	The internal Exam ID to open. Must be empty if **IntExceptionID** is not empty.
IntExceptionID	The internal Exception ID to open. Must be empty if **IntExamID** is not empty.
Reveal	True if this exam should be made visible. Revealing an exam will also make that Canvas Page visible.
Lock	True if the system should try to lock the exam upon creating the Canvas Page. An exam must be locked to mark it read.
OpenNew	True if an existing page should not be used.

Return Values

Value	Meaning
BSTR	String containing the **CanvasPageID** or empty if an error occurred

Error Codes

0,6,10,17,101,133, 1000

Go to Error Codes

Remarks

You can use **FindExam()** or **Query()** to find internal Exam IDs. If a Canvas Page for this exam or exception is already open and **OpenNew** is true, an error will be returned. If a Canvas Page for this exam or exception is already open and **OpenNew** is false, the **CanvasPageID** for that page will be returned. If **Lock** is TRUE, you should check to make sure the exam was in fact locked by calling **GetShelfStatus**.

See also

FindExam

Go to Table of Contents

2.2.18 GetCanvasPageStatus

This method is used to get the status for a Canvas Page.

```
BSTR GetCanvasPageStatus(
     BSTR CanvasPageID);
```

Parameters

Name	Description
CanvasPageID	The unique identifier for the Canvas Page.

The returns an XML string with the following elements and structure.

Name	Description
CanvasPageStatus	Contains the following elements:
PatientName	The name of the Patient
MRN	The MRN for this Patient
IntPatientID	The internal Patient ID for this Canvas Page

Example

```
<CanvasPageStatus>
<PatientName>Smith, Jane</PatientName>
<MRN>123456</MRN>
<IntPatientID>932</IntPatientID>
</CanvasPageStatus>
```

Error Codes

17

Go to Error Codes

Remarks

None.

See also

ListCanvasPages

Go to Table of Contents

2.2.19 CloseCanvasPage

This function closes the specified Canvas Page.

```
BSTR CloseCanvasPage(
     BSTR CanvasPageID.
     boolean DisgardChanges);
```

Parameters

Name	Description
CanvasPageID	The **CanvasPageID** of the Canvas Page to close.
DiscardChanges	If True, any changes made to this exam are lost without prompting the user.

Return Values

Value	Meaning
True	Success
False	Error, check with **GetLastErrorCode()**

Error Codes

0,6,7,10,17,125

Go to Error Codes

Remarks

None.

See also

ListCanvasPages

Go to Table of Contents

2.2.20 ListShelfs

This method is used to get the loaded shelves for a Canvas Page.

```
BSTR ListShelfs(
     BSTR CanvasPageID);
```

Parameters

Name	Description
CanvasPageID	The **CanvasPageID** for the Canvas Page.

Return Values

Value	Meaning	
BSTR	XML encoded string with shelf status. Includes the following attributes:	
	Shelfs	Collection of Shelfs for this Canvas Page
	ShelfID	The **ShelfID**’s for the Shelfs loaded into this Canvas Page

Error Codes

0,1,3,4,10,17,125

Go to Error Codes

Remarks

None.

See also

ListCanvasPages

Go to Table of Contents

2.2.21 OpenShelf

This function opens the specified Exam in a new Canvas Page

```
BSTR OpenShelf(
     BSTR CanvasPageID,
     BSTR IntExamID,
     BSTR IntExceptionID,
     boolean Reveal);
```

Parameters

Name	Description
CanvasPageID	The **CanvasPageID** to create the shelf in.
IntExamID	The internal Exam ID to open. Must be empty if **IntExceptionID** is not empty.
IntExceptionID	The internal Exception ID to open. Must be empty if **IntExamID** is not empty.
Reveal	True if this exam should be made visible. Revealing an exam will also make that Canvas Page visible.

Return Values

Value	Meaning
BSTR	String containing the Shelf ID or empty if an error occurred

Error Codes

0,6,10,17,101,125,1000

Go to Error Codes

Remarks

You can use **FindExam()** or **Query()** to find internal Exam IDs. If the shelf already exists for this exam or exception, the **ShelfID** of the already open shelf is returned. An exam can only be locked in the following case:

- Exam is not locked by another user
- Exam Status is Completed or InProgress.

See also

ListCanvasPages, FindExam

Go to Table of Contents

2.2.22 GetShelfStatus

This method is used to get the status for a shelf.

```
BSTR GetShelfStatus(
     BSTR ShelfID);
```

Parameters

Name	Description
ShelfID	The Shelf ID.

Remarks

The returns an XML string with the following elements and structure.

Name	Description
ShelfStatus	Contains the following elements:
CanvasPageID	The Canvas Page ID for the Canvas Page that owns this shelf
IntExamID	The internal Exam ID for this shelf (empty if this shelf manages an exception study)
IntExceptionStudyID	The internal exception Study ID for this shelf (empty if this shelf manages an exam)
MainExam	True/False indicates if this shelf manages the main exam for this Canvas Page
Locked	True/False indicates if the exam is locked by this user
x00080050	String containing the accession.
ExamReadFLAG	“Y” or “N” indicates if the exam is marked read.
IDXExamStatus	String containing the exam status. See Query function for definition

Example

```
<ShelfStatus>
<CanvasPageID>13128660683248<CanvasPageID>
<IntExamID>223</IntExamID>
<IntExceptionStudyID></IntExceptionStudyID>
<MainExam>1</MainExam>
<Locked>1</Locked>
<x00080050>2307755</x00080050>
<ExamReadFLAG>Y</ExamReadFLAG>
<IDXExamStatus>P</IDXExamStatus>
</ShelfStatus>
```

Error Codes

0,1,3,4,10,17,126

Go to Error Codes

Remarks

None.

See also

ListShelfs, OpenShelf

Go to Table of Contents

2.2.23 CloseShelf

This function closes the specified shelf on a Canvas Page.

```
BOOL CloseShelf(
      BSTR ShelfID);
```

Parameters

Name	**Description**
ShelfID	The Shelf ID for the shelf to close.

Return Values

Value	**Meaning**
True	Success
False	Error, check with **GetLastErrorCode()**

Error Codes

0,10,17,126,132

Go to Error Codes

Remarks

You cannot close the main exam for a Canvas Page.

See also

ListShelfs, OpenShelf

Go to Table of Contents

2.2.24 CopyToClipboard

This function copies the contents of the specified Image window to the clipboard.

```
boolean CopyToClipboard (
     BSTR WindowID);
```

Parameters

Name	Description
WindowID	Window identifier

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,17,120

Go to Error Codes

Remarks

The Window ID can be retrieved by trapping a View menu item event or by getting the active window. You may also get a list of the Window IDs for a loaded exam. See GetActiveWindow(), EventViewMenuSelected or GetExamWindowIDs() for more details.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.25 CopyImageToClipboard

This function copies the underlying image contents (no annotations or overlays) of a window to the clipboard.

```
  boolean CopyImageToClipboard(BSTR WindowID);
```

Parameters

Name	Description
WindowID	Window ID to copy image from.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Remarks

None.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.26 CopyWindowToPicture

This function returns a pointer to an IPicture object for the specified region of a window.

```
IDispatch *CopyImageToPicture(
     BSTR WindowID,
     long Left,
     long Top,
     long Right,
     long Bottom);
```

Parameters

Name	Description
WindowID	Window ID to create image from.
Left	Left pixel column of rectangle to create image from.
Top	Top pixel row of rectangle to create image from.
Right	Right pixel column of rectangle to create image from.
Bottom	Bottom pixel row of rectangle to create image from.

Return Values

Value	Meaning
IDispatch *	Pointer to an IPicture object.

Error Codes

17,120

Go to Error Codes

Remarks

A NULL pointer is returned in the event of an error.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.27 CopyImageToPicture

This function returns a pointer to an IPicture object for the specified region of the underlying image displayed in a given window (not including overlays or annotations).

```
IDispatch *CopyImageToPicture(
     BSTR WindowID,
     long Left,
     long Top,
     long Right,
     long Bottom);
```

Parameters

Name	Description
WindowID	Window ID to create image from.
Left	Left DICOM pixel column of rectangle to create image from.
Top	Top DICOM pixel row of rectangle to create image from.
Right	Right DICOM pixel column of rectangle to create image from.
Bottom	Bottom DICOM pixel row of rectangle to create image from.

Return Values

Value	Meaning
IDispatch *	Pointer to an IPicture object.

Error Codes

17,120

Go to Error Codes

Remarks

A NULL pointer is returned in the event of an error.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.28 GetShelfWindowIDs

This function returns a XML string containing a list of **WindowIDs** for the specified shelf. This function returns the **WindowIDs** of the rack windows only. It will not return the **WindowIDs** of any popup windows or fixed windows on the diagnostic monitors.

```
BSTR GetShelfWindowIDs (
     BSTR ShelfID);
```

Parameters

Name	Description
ShelfID	The Shelf ID to get Window IDs from.

Return Values

Value	Meaning
XML	List of Window IDs
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,10,17,126

Go to Error Codes

Remarks

This specified exam must already be open before calling this method.

Example

```
<WindowIDs>
<ID>123456</ID>
<ID>123457</ID> <ID>123458</ID>
</WindowIDs>
```

See also

ListShelfs, OpenShelf

Go to Table of Contents

2.2.29 GetWindowContext

This function retrieves contextual information about the specified window.

```
BSTR GetWindowContext(
      BSTR WindowID);
```

Parameters

Name	Description
WindowID	Window identifier

Return Values

Value	Meaning
XML	Contextual information about the specified window
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,10,17,120

Go to Error Codes

Remarks

The XML String contains the following elements:

Name	Description
WindowContext	Contains the following attributes
ImageUID	The UID of the image currently displayed
WindowWidth	The width of the window
WindowCenter	The center of the window
ZoomLevel	The zoom level
CenterPoint	The center point of this image
Rotation	Indicates the rotation 0=0, 1=90, 2=180, 3=270
Flipped	True if image has been flipped horizontally
Invert	True if this image has been inverted

Example

```
<WindowContext>
     <ImageUID>1.2.3.4.5.6.1</ImageUID>
     <WindowLevel>-185</WindowLevel>
     <ZoomLevel>2</ZoomLevel>
     <CenterPoint>100,200</CenterPoint>
     <Rotation>0</Rotation>
     <Flip>0</Flip>
     <Invert>0</Invert>
</WindowContext>
```

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.30 GetDICOMValue

This function retrieves a DICOM value from the specified image window.

```
BSTR GetDICOMValue(
      BSTR WindowID,
      BSTR DICOMTag);
```

Note: This function does not return any sequence level DICOM tag values.

Parameters

Name	Description
WindowID	Window identifier
DICOMTag	DICOM tag. Must be in hexadecimal form (for example, 0x00280030).

Return Values

Value	Meaning
BSTR	String representation of the DICOM value.
“”	Empty string if tag is not found or upon error. Use **GetLastErrorCode()** to determine error.

Error Codes

0,10,15,17,120

Go to Error Codes

Remarks

None.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.31 GetDICOMInstance

This function retrieves the DICOM instance from the specified window.

```
long GetDICOMInstance(
     BSTR WndID,
     BSTR ImageUID,
     VARIANT *DICOMData);
```

Parameters

Name	Description
WndID	Window identifier
ImageUID	The Image UID for which we retrieve the DICOMData.
DICOMData	The DICOMData formatted as a SAFEARRAY of bytes.

Return Values

Value	Meaning
long	The size of the DICOM Data

Error Codes

0,10,17,120,121, 227

Go to Error Codes

Note: This method is only available if the **WorkstationLocation** is “Main Location” at the time of user login. This method is not supported for Multi-Frame DICOM images. Use WriteDicomInstance instead.

Remarks

The **SAFEARRAY** that’s being generated starts on index 1 for compatibility with scripting languages.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.32 WriteDICOMInstance

This function retrieves the DICOM instance from the specified window and writes it to the specified file in DICOM Part 10 format.

```
boolean WriteDICOMInstance(
      BSTR WindowID,
      BSTR ImageUID,
      BSTR PathName);
```

Parameters

Name	Description
WindowID	Window identifier
ImageUID	The Image UID for which we retrieve the DICOMData.
PathName	The full path and filename of the file to write to.

Return Values

Value	Meaning
True	Success
False	Failure, check error with **GetLastErrorCode()**

Error Codes

0,10,17,120,121

Go to Error Codes

Note: This method is only available if the **WorkstationLocation** is **Main Location**.

Remarks

If the file does not exist on the local machine, this function will create it. If the file already exists, this function will overwrite any existing data.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.33 GetDICOMHeaders

This function returns a DICOM Part 10 header as a byte array.

```
long GetDICOMHeaders(
     BSTR StudyID,
     BSTR SeriesID,
     BSTR ImageID,
     VARIANT *DICOMData);
```

Parameters

Name	Description
StudyID	The DICOM Study Instance UID.
SeriesID	The DICOM Series Instance UID.
ImageID	The DICOM Image Instance UID.
DICOMData	The DICOMData formatted as a SAFEARRAY of bytes.

Return Values

Value	Meaning
long	Size of the DICOM data.

Error Codes

17,105, 106, 121, 1000

Go to Error Codes

Remarks

This function returns the DICOM Part 10 header as a byte array. A zero length buffer is returned in the event of an error.

Note: This method can only be used if the image is currently hung in a Canvas Page.

Go to Table of Contents

2.2.34 GetDICOMPixels

Returns the pixel data for an image as an array of either shorts or longs depending on the image.

```
long GetDICOMPixels(
     BSTR bstrStudyID, BSTR bstrSeriesID, BSTR bstrImageID,
     long nLeft, long nTop, long nRight, long nBottom,
     long nCompressionRatio,
     VARIANT *pvarDICOMData);
```

Parameters

Name	Description
bstrStudyID	The DICOM Study Instance UID.
bstrSeriesID	The DICOM Series Instance UID.
bstrImageID	The DICOM Image Instance UID.
nLeft	Left pixel column of the image rectangle.
nTop	Top pixel row of the image rectangle.
nRight	Right pixel column of the image rectangle.
nBottom	Bottom pixel row of the image rectangle.
nCompressionRatio	Image Compression Ratio.
pvarDICOMData	The DICOMData formatted as a SAFEARRAY.

Return Values

Value	Meaning
long	Size of the DICOM data.
VARIANT	One successful return pvarDICOMData is loaded with Pixel data

Remarks

The data returned will be formatted as a **SAFEARRAY**. The array will contain either 4-byte RGB values for color or 16-bit integers in gray-scale irrespective of what was received from the modality system, indicated in the header data. The integrating system should interpret the data accordingly.

The data returned includes the entire rectangle described by the left, top, right and bottom coordinates. That is, the coordinates are inclusive. The specified rectangle is interpreted as to be given in compressed image coordinates.

If the coordinates don’t describe a rectangle completely within the compressed image, an error code is returned. However, if **nRight** or **nBottom** are specified as -1, the images right and bottom, respectively, edges are used instead.

If the requested DICOM image is a multiframe image, it is necessary to append the frame number at the end of the **ImageID** string in the format of "-x", where x is the number of the frame to return the pixel data for. Frame numbering starts with 1.

For example, to request the 14th frame from ImageID "1.2.3.4", use "1.2.3.4-14" for the **ImageID** parameter.

To check if an image is a multiframe image, use the **GetDICOMValue** function to check for the tag value "0x00280008". If the image is a multiframe image, this tag will return the number of frames in the multiframe image. Non-multiframe images will return an empty string and **GetLastErrorCode** will return error code **15**.

Note: This method can only be used if the image is currently hung in a Canvas Page.

Go to Table of Contents

2.2.35 DisplayMediaExportPage

This function adds exams to the Media Export Page, and displays the Media Export Page in a dialog window.

```
boolean DisplayMediaExportPage(
      BSTR InternalExamIDs,
      boolean RemoveAllExams);
```

Parameters

Name	Description

InternalExamIDs	XML list of internal exams ID. For example:
	`<IntExamIDs>`
	    `<IntExamID>…</ IntExamID >`
	    `<IntExamID>…</ IntExamID >`
	    `<IntExamID>…</ IntExamID >`
	`</IntExamIDs>`

RemoveAllExams	If True, removes all previously added exams to the export page.

Return Values

Value	Meaning
True	Success
False	Failure, check error with **GetLastErrorCode()**

Error Codes

0,10,17,19, 20

Go to Error Codes

Remarks

This function is only available if the **HideFolder** option is enabled.

Note: **DisplayMediaExportPage** can only be used to add non-exception studies to the Media Export Page.

See also

FindExam

Go to Table of Contents

2.2.36 SetWindowImage

This function changes the Image displayed in the current window.

```
boolean SetWindowImage(
      BSTR WindowID,
      BSTR ImageUID);
```

Parameters

Name	Description
WindowID	The Window ID
ImageUID	The UID for the image to display in that window.

Return Values

Value	Meaning
True	Success
False	Failure, check error with **GetLastErrorCode()**

Error Codes

0,7,10,17,120,121

Go to Error Codes

Remarks

This function is used to change the current image for a stack window. **ImageUID** must be one of the images in the stack otherwise, this function will fail. If the specified window is not currently visible in the rack, the rack is scrolled so it is.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.37 SetWindowView

This function sets the window's zoom level and top left corner. This function is only valid for popup windows.

```
boolean SetWindowView (
BSTR WindowID,
short ZoomLevel,
short Top,
short Left,
long WindowWidth,
long WindowCenter);
```

Parameters

Name	Description
WindowID	The Window ID.
ZoomLevel	The level to zoom the image to. Zoom level must be greater than 0.
Top	The top row to display in full resolution coordinates.
Left	The left column to display in full resolution coordinates.
WindowWidth	The width of the histogram window.
WindowCenter	The center of the histogram window.

Return Values

Value	Meaning
True	Success
False	Failure, check error with **GetLastErrorCode()**

Error Codes

0,4,10,17,120,122,123

Go to Error Codes

Remarks

This function does not return until the window has been fully updated. If the specified window is not currently visible in the rack, the rack will be scrolled to make it visible.

Note: This function is only valid for popup windows created through the **CreatePopup** method.

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.38 GetStaticWindowInfo

This function gets the static (non changing) information about the image in the specified window.

```
BSTR GetStaticWindowInfo(
     BSTR WindowID);
```

Parameters

Name	Description
WindowID	The Window ID

Return Values

Value	Meaning
XML	String containing the static information for the specified window
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,10,17,120

Go to Error Codes

Remarks

The XML String contains the following elements:

Name	Description
Window	Contains the following attributes
Popup	0 = Rack image, 1 = popup window
ImageHung	1 = Image is hung on a diagnostic monitor
hWnd	The Windows handle for this window
CanvasPageID	The Canvas Page ID for the Canvas Page that manages this window
ShelfID	The Shelf ID for the shelf that manages this window
StudyUID	The DICOM Study Instance UID
SeriesUID	The DICOM Series Instance UID
MaxLevels	The number of transformation levels
Rows	Number of Rows in this image
Columns	Number of Columns in this image
ColPixelSpacing	The Column pixel spacing (width of each pixel in mm)
RowPixelSpacing	The Row pixel spacing (height of each pixel in mm)

Name	Description
Modality	The modality for this window
ImageUIDs	Contains a list of image UIDs in this window
UID	An Image UID

Example

```
<Window>
<Popup>0</Popup>
<ImageHung>1</ImageHung>
<hWnd>34562334</hWnd>
<IntExamID>11111</IntExamID>
<StudyUID>1.2.3.4</StudyUID>
<SeriesUID>1.2.3.4.1</SeriesUID>
<MaxLevels>3</MaxLevels>
<Rows>512</Rows> <Columns>512</Columns>
<ColPixelSpacing>0.0723</ColPixelSpacing>
<RowPixelSpacing>0.0723</RowPixelSpacing>
<Modality>CT</Modality>
<ImageUIDs>
<UID>1.2.3.4.1.1</UID>
<UID>1.2.3.4.1.2</UID>
<UID>1.2.3.4.1.3</UID>
<UID>1.2.3.4.1.4</UID> </ImageUIDs>
</Window>
```

See also

GetActiveWindow, EventViewMenuSelected

Go to Table of Contents

2.2.39 GetActiveWindow

This function returns the Window ID of the currently active window.

```
  BSTR GetActiveWindow();
```

Return Values

Value	Meaning
WindowID	The Window ID of the active window
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,10,17

Go to Error Codes

Remarks

The currently active window is the image or thumbnail window currently under the cursor. Note that if the cursor is over an image in the rack it will return the Window ID for that image and not any of the corresponding images on the monitors or in popup windows. This function will return an empty string if the cursor is moved from an image into the Timeline or other parts of the rack.

Go to Table of Contents

2.2.40 MarkExamRead

This function marks the exam as read, saves a Mark Read Presentation State, and closes the Canvas Page containing the exam. This function mirrors the behavior of the Mark Read button in the iSite Radiology user interface.

```
BOOL MarkExamRead(
      BSTR InternalExamID);
```

Parameters

Name	Description
InternalExamID	The Internal Exam ID of the exam to mark read.

Return Values

Value	Meaning
BOOL	True = success; False = failure
False	Use **GetLastErrorCode()** to determine error

Error Codes

0,6,7,9,17,129,130,131,1000

Go to Error Codes

Remarks

This function must be called after successful completion of **OpenCanvasPage()**.

See also

FindExam

Go to Table of Contents

2.2.41 SetMarkRead

This function marks exam as Read or Unread.

```
BOOL SetMarkRead (BSTR IntExamID, BOOL ReadFlg);
```

Parameters

Name	Description
IntExamID	The internal Exam ID.
ReadFlg	True = mark exam Read, False- mark exam Unread.

Return Values

Value	Meaning
BOOL	True = success; False = failure.
False	Use **GetLastErrorCode()** to determine error.

Error Codes

0,2,6,9,10,17,101,1000

Go to Error Codes

Remarks

- A lock on the exam is required in order to mark the exam as Read/Unread.
- The exam Read flag is independent of the actual exam status. You can mark a scheduled exam as Read or a finalized exam as Unread.
- If an exam was marked as Read or Unread already, function returns True.

Note: This function is equivalent to iSite Enterprise control's **MarkExamRead()** function.

See also

FindExam

Go to Table of Contents

2.2.42 GetCurrentUser

This function will return the current user who is logged in.

```
  BSTR GetCurrentUser();
```

Return Values

Value	Meaning
BSTR	String containing the current User Name. This string will be empty if no user is logged in.

Error Codes

0,2,10

Go to Error Codes

Remarks

None.

Go to Table of Contents

2.2.43 AddPreferencePage

This method is used to add a User, System, or Machine Preference page to the Preferences dialog box.

```
boolean AddPreferencePage(
      BSTR Name,
      BSTR URL,
      BSTR Type);
```

Parameters

Name	Description
Name	The node name for this Preference page.
URL	The URL to load for this Preference page.
Type	One of:

Name	Description
User	Creates the page under the User preference node
System	Creates the page under the System preference node
Machine	Creates the page under the Machine preference node

Return Values

Value	Meaning
True	Success
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

0,10,17

Go to Error Codes

Remarks

AddPreferencePage is called from a plug in page, to configure the plug in, if required.

Go to Table of Contents

2.2.44 EnablePreferenceApplyButton

This method is used to enable the Apply button in the Preference dialog box.

```
boolean EnablePreferenceApplyButton();
```

Parameters

None.

Return Values

Value	Meaning
True	Success
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

0,10

Go to Error Codes

Remarks

This function should only be called from a page added by **AddPreferencePage**, and be used to enable the Apply button after the user has modified a preference. It allows the coder of the Preference page to check for valid preference values before the preference can be applied by the user.

Go to Table of Contents

2.2.45 MessageBox

This method is used to display an **AfxMessageBox** style message dialog box.

```
int MessageBox(
      BSTR MessageText,
      int MessageType);
```

Parameters

Name	Description
MessageText	Message to be displayed in the message box.
MessageType	Type of message box (see **AfxMessageBox**)

Return Values

Value	Meaning
Integer	See AfxMessageBox

Error Codes

0,10,17

Go to Error Codes

Remarks

Use this instead of, for example an alert message, so that the message box window will have the intended z order and not be hidden by other windows.

Go to Table of Contents

2.2.46 SetActivePage

This method is used to change the active page.

```
boolean SetActivePage(
      BSTR Name,
      BSTR Type);
```

Parameters

Name	Description
Name	The node name for the active page.

Type	FOLDER = Folder tab
	CANVAS = Canvas tab
	API = API Specified tab

Return Values

Value	Meaning
True	Success
False	Failure, check error code with **GetLastErrorCode()**

Error Codes

0,4,10,17

Go to Error Codes

Remarks

FOLDER: Because the folder page structure is hierarchical, the full path is included in the name to distinguish between pages of the same name.

Example: `SetActivePage("iSite Tools\Patient Directory", "FOLDER")`

`SetActivePage("Public Folders\Interesting Cases", "FOLDER")`

CANVAS: When changing to a Canvas Page, the name parameter is the *CanvasPageID* of that Canvas Page.

Example: `SetActivePage("123456789", "CANVAS")`

API: API is used for navigating to any iSite Plug-In Page.

Example: `SetActivePage("iSite Tools\My Custom Plugin", "API")`

Note: If using a C-style language, you must use the backslash escape sequence "\\" in the *Name* parameter.

Go to Table of Contents

2.2.47 FindShelfID

This method is used to get the Shelf IDs for an internal Exam ID.

```
BSTR FindShelfID(
     BSTR IntExamID);
```

Parameters

Name	Description
IntExamID	The internal Exam ID for the shelf.

Return Values

Value	Meaning	
BSTR	XML encoded string with Shelf IDs. Includes the following attributes:	
	ShelfIDs	Collection of Shelfs for this internal Exam ID
	ID	The ShelfID's for the shelf internal Exam ID

Error Codes

0,1,3,4,10,17,126

Go to Error Codes

Remarks

None.

Sample

```
<ShelfIDs>
     <ID>46132014090831</ID>
</ShelfIDs>
```

See also

FindExam

Go to Table of Contents

2.2.48 FindCanvasPageID

This method is used to get the Canvas Page IDs for an internal Patient ID.

```
BSTR FindCanvasPageID(
     BSTR IntPatientID);
```

Parameters

Name	Description
IntPatientID	The internal Patient ID for the Canvas Page.

Return Values

Value	Meaning	
BSTR	XML encoded string with Canvas Page IDs. Includes the following attributes:	
	CanvasPageIDs	Collection of Canvas Pages for this internal Patient ID
	ID	The ShelfID's for the shelf internal Exam ID

Error Codes

0,1,3,4,10,17,125

Go to Error Codes

Remarks

None.

Sample

```
<CanvasPageIDs>
     <ID>46133014090833</ID>
</ CanvasPageIDs >
```

See also

FindPatient

Go to Table of Contents

2.2.49 ListMediaExportExams

This method returns the list of all the exams in XML format which the user added to the CD Manager folder for exporting.

```
BSTR ListMediaExportExams();
```

Parameters

None.

Return Values

Value	Meaning
BSTR	XML encoded string. Includes the following attributes shown below.

Example

```
<MediaExportExamAttributes>
    <ExamCount>"Number of Exams in the Media Export Folder"</ExamCount>
    <Exam1>
        <PatientName> "Patient Name associated with the exam"</PatientName>
        <MRN>"MRN of the exam"</MRN>
        <DOB>"Date of Birth of the exam"</DOB>
        <StudyDateTime>"Date Time of the study"</StudyDateTime>
        <Modality>"Modality of the exam"</Modality>
        <Accession>"Accession number of the exam"</Accession>
        <ReferingPhysician>"Refering physician of the exam"</ReferingPhysician>
        <ProcedureDescription>"Description of the exam" </ProcedureDescription>
        <BodyPart>"Body part of the exam"</BodyPart>
    </Exam1>
</MediaExportExamAttributes">
```

Error Codes

0, 205,206,1000

Go to Error Codes

Remarks

None.

Go to Table of Contents

2.2.50 AddShelfButton

This function adds a new Shelf button.

```
boolean AddShelfButton(
      BSTR ShelfID, BSTR ButtonID, BSTR BitMapName,
      BSTR ToolTip);
```

Parameters

Name	Description
ShelfID	The Shelf ID to which to add this button.
ButtonID	The unique Name of the button.
BitMapName	The name of the File on disk of the bitmap.
ToolTip	The Tooltip that is displayed for the button.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,17,110,111

Go to Error Codes

Remarks

Use this method to add a custom Shelf button. The **ButtonID** is not shown to the user, but is used to detect the button was pressed in the **EventShelfButton**. When the button is pressed the **EventShelfButton** event will be fired. The bitmap must exist, with a width of 20 pixels and a height of 20 pixels.

See also

FindShelfID, ListShelfs

Go to Table of Contents

2.2.51 ShowPreferenceDialog

This function launches the iSite PACS Preferences dialog box.

```
boolean ShowPreferenceDialog();
```

Parameters

None.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,9,10,216

Go to Error Codes

Remarks

Use this method to display the Preferences dialog box to the user. This method only functions if the **IDXMode** option is set.

Go to Table of Contents

2.2.52 EmergencyAccessLogin

This function attempts to log the user into the Emergency Access server. This is the same functionality as using the Emergency Access box on the login page.

```
boolean EmergencyAccessLogin(BSTR UserName, BSTR Password, BSTR AuthSource, BSTR
Token, BSTR Mnemonic);
```

Parameters

Name	Description
UserName	The user name.
Password	The password.
AuthSource	Contains either iSite PACS (same authentication as before) or IDXNTLM for NT domain authentication.
Token	A string passed to the authorizing server.
Mnemonic	A name uniquely identifying this user account. May be an empty string. This name is used for logging purposes only.

Return Values

Value	Meaning
True	The user was logged in.
False	The user was not logged in, check error code with **GetLastErrorCode()**.

Error Codes

0,5,6,7,8,9,17

Go to Error Codes

Remarks

If a user is currently logged in, you must logout before calling this function. The **Mnemonic** may be used to identify a user while logging into iSite Radiology using a single user name and password.

Go to Table of Contents

2.2.53 CacheExam

This function schedules an Exam for loading onto the local disk.

```
Boolean CacheExam(BSTR bstrIntExamID, BSTR bstrExcpID, VARIANT_BOOL bLockExam);
```

Parameters

Name	Description
bstrIntExamID	The internal Exam ID of the exam to be cached.
bstrExcpID	The ID of the exception to be cached.
bLockExam	Indicates if the exam is to be locked.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Remarks

Only **bstrIntExamID** or **bstrExcpID** should be specified, if both are given, **bstrExcpID** is ignored. These IDs can be found using the **FindExam** method.

See also

FindExam

Go to Table of Contents

2.2.54 ResumeCachingExam

This function restarts caching of an Exam that halted before completion.

```
Boolean ResumeCachingExam(BSTR bstrIntExamID, BSTR bstrExcpID);
```

Parameters

Name	Description
bstrIntExamID	The internal Exam ID of the exam to be cached.
bstrExcpID	The ID of the exception to be cached.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Remarks

Only **bstrIntExamID** or **bstrExcpID** should be specified, if both are given, **bstrExcpID** is ignored. These IDs can be found using the **FindExam** method.

See also

FindExam

Go to Table of Contents

2.2.55 CancelExamCaching

This function cancels the caching of an exam that was earlier started with **CacheExam**.

```
CancelExamCaching(BSTR bstrIntExamID, BSTR bstrExcpID);
```

Parameters

Name	Description
bstrIntExamID	The internal Exam ID of the exam to be cached.
bstrExcpID	The ID of the exception to be cached.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Remarks

Only **bstrIntExamID** or **bstrExcpID** should be specified, if both are given, **bstrExcpID** is ignored. These IDs can be found using the **FindExam** method.

See also

FindExam

Go to Table of Contents

2.2.56 DeleteCachedExam

This function deletes an exam that was earlier successfully loaded.

```
  DeleteCachedExam(BSTR bstrIntExamID, BSTR bstrExcpID);
```

Parameters

Name	Description
bstrIntExamID	The internal Exam ID of the exam to be cached.
bstrExcpID	The ID of the exception to be cached.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Remarks

Only **bstrIntExamID** or **bstrExcpID** should be specified, if both are given, **bstrExcpID** is ignored. These IDs can be found using the **FindExam** method.

See also

FindExam

Go to Table of Contents

2.2.57 GetCachedExams

This function returns a list of all Exams that have been cached locally.

```
BSTR GetCachedExams();
```

Parameters

None.

Return Values

Value	Meaning
BSTR	XML string of exam details.
“”	Failure, check for error with **GetLastErrorCode()**

Error Codes

None.

Go to Error Codes

Remarks

This function returns an XML string that details the list of all cached exams.

```
<CachedExams>
     <CachedExam>
            <ExamKey>InternalExamID</ExamKey>
            <ExceptionKey>InternalExceptionID</ExceptionKey>
            <Status>Cached Status</Status>
            <Progress>Percent of Exam Cached</Progress>
            <Size>Exam Size</Size>
            <Sender>Last, First name of user </Sender>
            <ReceivedOn>MM/DD/YYYY HH:MM:SS.mmmm</ReceivedOn>
     </CachedExam>
</CachedExams>
```

Go to Table of Contents

2.2.58 GetWorkstationLocations

This method is used to get the list of the defined workstation locations. In multi-site configurations it is important that your client application log into the appropriate location to optimize performance and network bandwidth.

```
BSTR GetWorkstationLocations();
```

Parameters

None.

Return Values

Value	Meaning	
BSTR	XML encoded string with Location Names:	
	WorkstationLocations	Collection of workstation locations for this server
	LocationName	The name of the location, i.e. “Wide Area Network”

Error Codes

0,2,6,7

Go to Error Codes

Remarks

To set the workstation location value, directly set the **Radiology.WorkstationLocation** property to the appropriate value.

For example: **`Radiology.WorkstationLocation = "Teleradiology"`**

Go to Table of Contents

2.2.59 GetAuthSources

This method is used to get the list of the defined authentication sources.

```
BSTR GetAuthSources();
```

Parameters

None.

Return Values

Value	Meaning	
BSTR	XML encoded string with Authentication Sources:	
	AuthSources	Collection of authentication sources for this server
	Name	The name of authentication source, i.e. “IDXrad”
	DisplayName	The user friendly name of authentication source, i.e. “iSite”

Error Codes

0,2,6,7

Go to Error Codes

Remarks

This value is used to get the non-visual control Authentication Source values after the non-visual control is initialized. If using an Authorization Source other than ‘iSite’ to log in, you must use the **Name** string in the **Login()** method.

Go to Table of Contents

2.2.60 DisplayiExportQueue

This function opens the iExport Queue window.

```
boolean DisplayiExportQueue();
```

Parameters

None.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Remarks

None.

Go to Table of Contents

2.2.61 DisableAutologout

This function enables or disables the iSite PACS Auto Logout feature.

```
void DisableAutologout(BOOL bDisable);
```

Parameters

Name	Description
bDisable	**True** to prevent iSite Radiology from logging out when idle, **False** to allow it to log out.

Return Values

None.

Remarks

None.

Go to Table of Contents

2.2.62 ShowDebugWindow

This function opens iSite's debug window.

```
boolean ShowDebugWindow();
```

Parameters

None.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

9

Go to Error Codes

Remarks

The iSite PACS debug window is useful for diagnosing problems with custom iSite Enterprise integrations.

Go to Table of Contents

2.2.63 GetFoldersAndFiltersXML

This function returns information about the Folders and Filters.

```
BSTR GetFoldersAndFiltersXML(Long Level);
```

Parameters

Name	Description
Level	0 = User Folder and Filters 1 = System Folder and Filters

Return Values

Value	Meaning
XML	See example below for format

Remarks

This method returns an XML string that contains a list of all folders and filters depending on the Level passed in.

- 0 returns all Folders and Filters associated with a specific user.
- 1 returns all system Folders and Filters.

Go to Table of Contents

2.2.64 GetVersion

This function returns the Version of iSite Radiology.

```
BSTR GetVersion();
```

Parameters

None.

Return Values

Value	Meaning
Version	Version information in the form “major.minor.revision.build”

Remarks

None.

Go to Table of Contents

2.2.65 FindException

This function queries the database for the internal Exception ID given a DICOM Study Instance UID.

```
BSTR FindException(BSTR StudyUID);
```

Parameters

Name	Description
StudyUID	The DICOM Study Instance UID.

Return Values

Name	Meaning
String	The internal exception Study ID
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,2,6,10,17

Go to Error Codes

Remarks

In some cases, a **StudyUID** may map to more than one Exception. In this case, this method will return one of the matching **InternalExceptionID’s**. If no matching exceptions are found, an empty string is returned and the error code is set to 0 (no errors).

Go to Table of Contents

2.2.66 DeleteAnnotation

This function deletes an Annotation from the specified window.

```
boolean DeleteAnnotation(
        BSTR WindowID,
        BSTR Token);
```

Parameters

Name	Description
WindowID	WindowID from which to remove the Annotation.
PresentationStateID	Annotation Token. (Annotation ID)

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

120, 220

Go to Error Codes

Remarks

The Token is obtained from the Annotation events.

Note: In iSite Radiology, Annotations are not visible in thumbnail windows. Because of this, in iSite Radiology, this method can only be called using a **WindowID** from a popup window or a diagnostic window.

See also

GetActiveWindow, IntroductionEventViewMenuSelected

Go to Table of Contents

2.2.67 SavePresentationState

This function creates a Presentation State based on the state of the Shelf used for Shelf ID.

```
BSTR SavePresentationState(
     BSTR ShelfID,
     BSTR PSDescription,
     int Type);
```

Parameters

Name	Description
ShelfID	Shelf ID from which to create the Presentation State.
PSDescription	Description displayed in the Presentation State Shelf context menu.
Type	Int referring to the type of Presentation State.

PS Type	Meaning
0	RawDicom
1	Technologist
2	Radiologist
3	PreRead
4	User

Return Values

Value	Meaning
BSTR	Presentation State ID of the newly created Presentation State
“”	Failure, check for error with **GetLastErrorCode()**

Error Codes

17, 126

Go to Error Codes

Remarks

None.

See also

FindShelfID, ListShelfs

Go to Table of Contents

2.2.68 LoadPresentationState

This function applies a saved Presentation State into the specified shelf.

```
boolean LoadPresentationState(
      BSTR ShelfID,
      BSTR PresentationStateID);
```

Parameters

Name	Description
ShelfID	Shelf ID to which to load the Presentation State.
PresentationStateID	Presentation State ID of the Presentation State to load.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

17, 126

Go to Error Codes

Remarks

The **PresentationStateID** can be obtained from either the **SavePresentationState** method or the **PresentationState** events.

See also

FindShelfID, ListShelfs, SavePresentationState

Go to Table of Contents

2.2.69 ShowiQueryWindow

This function opens the iQuery.

```
boolean ShowiQueryWindow(BSTR IntPatientID);
```

Parameters

Name	Description
IntPatientID	IntPatientID from which to populate the iQuery exams list.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

9, 17, 222

Go to Error Codes

Remarks

None.

See also

FindPatient

Go to Table of Contents

2.2.70 CreatePopup

This function creates popup window based on navigation Window ID.

```
BSTR CreatePopup(
      BSTR WindowID);
```

Parameters

Name	Description
WindowID	Navigation Window ID.

Return Values

Value	Meaning
BSTR	Popup Window ID
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,2,10,17,120

Go to Error Codes

Remarks

For this function to succeed, exam has to be opened and revealed. **CreatePopup** will return either the new Window ID or any existing one.

See also

GetActiveWindow, IntroductionEventViewMenuSelected

Go to Table of Contents

2.2.71 DestroyPopup

This function destroys popup window based on popup Window ID.

```
  BOOL DestroyPopup(BSTR WindowID);
```

Parameters

Name	Description
WindowID	Popup Window ID.

Return Values

Value	Meaning
BOOL	True = success; False = failure
False	Use **GetLastErrorCode()** to determine error

Error Codes

0,2,10,17,120

Go to Error Codes

Remarks

For this function to succeed, a popup window has to be created.

See also

GetActiveWindow, IntroductionEventViewMenuSelected

Go to Table of Contents

2.2.72 ShowExportViaDICOMWindow

This function opens the Export via DICOM dialog box.

```
boolean ShowExportViaDICOMWindow(
      BSTR IntExamID,
      BSTR IntExcpID,
      BOOL CheckAll);
```

Parameters

Name	Description
IntExamID	Internal Exam ID of the exam that should be checked for export by default.
IntExcpID	Internal Exception ID of the exam that should be checked for export by default. (Ignored if **IntExamID** is specified).
CheckAll	True to check all exams for the patient when the dialog box is launched.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Remarks

Only one of the **IntExamID** or **IntExcpID** parameters should be specified. An empty string should be passed in for the unspecified parameter. If both parameters are non-empty strings, **IntExcpID** is ignored.

This method opens the Export via DICOM dialog which lists all exams associated with the patient that the specified exam belongs to. By default, the exam passed in through **IntExamID** or **IntExcpID** is checked.

See also

FindExam

Go to Table of Contents

2.2.73 LockExam

This function locks or unlocks an exam.

```
  boolean LockExam(BSTR IntExamID, BOOL Lock);
```

Parameters

Name	Description
IntExamID	The internal Exam ID of the exam to lock.
Lock	True = lock exam; False = unlock exam.

Return Values

Value	Meaning
BOOL	True = success; False = failure
False	Use **GetLastErrorCode()** to determine error

Error Codes

0,2,10,16,101,127,128,129,1000

Go to Error Codes

Remarks

LockExam() does not validate **IntExamID** except the case when it is an empty string. If an empty string is passed in, error code 101 is returned from **GetLastErrorCode()**.

See also

FindExam

Go to Table of Contents

2.2.74 FindStudiesFromExam

This method is used to get list of all the studies related to the specified exam.

```
BSTR FindStudiesFromExam(
     BSTR ExamID);
```

Parameters

Name	Description
ExamID	The ID of the exam for which studies will be returned.

Return Values

Value	Meaning
BSTR	XML String containing the list of studies. The string is empty if an error occurred.

Error Codes

0,1000

Go to Error Codes

Remarks

This function returns an XML string that lists of all studies for the specified exam.

```
<StudyList>
     <StudyID>1.2.3.40000.50000</StudyID>
     <StudyID>1.2.3.40000.50001</StudyID>
     <StudyID>1.2.3.40000.50002</StudyID>
</StudyList>
```

See also

FindExam

Go to Table of Contents

2.2.75 FindExamsFromStudy

This method is used to get a list of all exams and exceptions related to the specified study.

```
BSTR FindExamsFromStudy (
     BSTR StudyID);
```

Parameters

Name	Description
StudyUID	The ID of the study for which exams and/or exceptions will be returned.

Return Values

Value	Meaning
BSTR	XML String containing the list of exam and/or exceptions. The string is empty if an error occurred.

Error Codes

0,1000

Go to Error Codes

Remarks

This function returns an XML string that lists of all exams for the specified study.

```
<ExamList>
     <ExamID>9876</ExamID>
     <ExamID>9877</ExamID>
     <ExceptionID>9889</ExceptionID>
</ExamList>
```

Go to Table of Contents

2.2.76 FindLinkedExams

This method is used to get a list of all exams linked to the specified exam. A linked exam is one that is related to the same study as the specified exam.

```
BSTR FindLinkedExams (
     BSTR ExamID);
```

Parameters

Name	Description
ExamID	The ID of the exam for which exams will be returned.

Return Values

Value	Meaning
BSTR	XML String containing the list of the linked exams. The string is empty if an error occurred.

Error Codes

0,17, 110, 1000

Go to Error Codes

Remarks

This function returns an XML string that lists of all exams linked to the specified exam.

```
<ExamList>
     <ExamID>9876</ExamID>
     <ExamID>9877</ExamID>
     <ExamID>9878</ExamID>
</ExamList>
```

The ExamID input parameter will appear in the XML output.

See also

FindExam

Go to Table of Contents

2.2.77 GetAvailableLanguages

This function returns an XML stream describing the languages installed on the iSite PACS client application. This is only available in iSite PACS 4.x.

```
BSTR GetAvailableLanguages();
```

Parameters

None.

Return Values

Value	Meaning
BSTR	XML string of all available languages.

The returns an XML string with the following elements and structure.

Name	Description
AvailableLanguages	Contains active language and all languages the system supports with code, name, and Language ID
Code	Language Code
Name	Name of the language
LCID	Internal Language ID

Example:

```
<Languages>
     <Language>
     <Active>
     <Code>ENG</Code >
     <Name>English</Name>
     <LCID>1033</LCID>
     </Active>
     </Language>
     <Language>
     <Code>DEU</Code >
     <Name>German</Name>
     <LCID>1031</LCID>
     </Language>
</Languages>
```

Remarks

All available languages that system supports are returned. The return string is in English.

Go to Table of Contents

2.2.78 SetLanguage

Specifies which language iSite Radiology will show.

```
boolean SetLanguage(long LCID);
```

Parameters

Name	Description
LCID	Identifies a locale for national language support. Locale information is used for international string comparisons and localized member names.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,5,225

Go to Error Codes

Remarks

The LCID must be one of those returned by **GetAvailableLanguages**. **SetLanguage** must be called before the user logs in.

See also

GetAvailableLanguages

Go to Table of Contents

2.2.79 AddViewMenuItem

This function adds a new menu item to the Image menu.

```
boolean AddViewMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,111

Go to Error Codes

Remarks

Use this method to add up to five menu items to the Image context menu. When one of these menu items is selected, the **EventMenuSelected** event will be fired.

Go to Table of Contents

2.2.80 RemoveViewMenuItem

This function removes a previously added menu or submenu from the Image menu.

```
boolean RemoveViewMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110

Go to Error Codes

Remarks

Use this method to remove a menu item previously added with **AddViewMenuItem** or **InsertAfterViewMenuItem**.

See also

AddViewMenuItem

Go to Table of Contents

2.2.81 InsertAfterViewMenuItem

This function adds new menu item to the Image menu after the specified menu item.

```
boolean InsertAfterViewMenuItem (
      BSTR Name,
      BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of new menu item, must be unique.
NameAfter	Name to insert menu item after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to insert a menu item after another previously added menu item.

See also

AddViewMenuItem

Go to Table of Contents

2.2.82 AddViewSubMenu

This function adds a new submenu item to the Image menu.

```
boolean AddViewSubMenu(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add up a submenu to the Image context menu. Items are added below it with **AddViewSubMenuItem()**.

Go to Table of Contents

2.2.83 AddViewSubMenuItem

This function adds a new submenu item to the Image submenu.

```
boolean AddViewSubMenuItem(
      BSTR Name, BSTR SubMenu);
```

Parameters

Name	Description
Name	Name of menu item, must be unique
SubMenu	Name of submenu this item belong to

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenu items to the Image context menu. A submenu must first be added with **AddViewSubMenu()**. When one of these menu items is selected, the **EventViewMenuSelected** event will be fired.

See also

AddViewSubMenu

Go to Table of Contents

2.2.84 InsertAfterViewSubMenu

This function adds a new submenu to the Image menu, after the menu item given.

```
boolean InsertAfterViewSubMenu(
     BSTR Name, BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
NameAfter	Name of menu item this submenu should be added after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenus to the Image context menu, after an existing item.

See also

AddViewSubMenu

Go to Table of Contents

2.2.85 InsertAfterViewSubMenuItem

This function adds a new submenu item to the Image menu, after the submenu item given.

```
boolean InsertAfterViewSubMenuItem(
      BSTR Name, BSTR SubMenu, BSTR SubNameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
SubMenu	Name of submenu to which this item belongs.
NameAfter	Name of menu item this submenu should be added after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenu items to the Image context menu, after an existing submenu item.

See also

AddViewSubMenu, AddViewSubMenuItem

Go to Table of Contents

2.2.86 AddExamMenuItem

This function adds a new menu item to the Exam menu.

```
boolean AddExamMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add up to five menu items to the Exam menu. When one of these menu items is selected, the **EventExamMenuSelected** event will be fired.

Go to Table of Contents

2.2.87 RemoveExamMenuItem

This function removes a previously added menu or submenu from the Exam menu.

```
boolean RemoveExamMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110

Go to Error Codes

Remarks

Use this method to remove a menu item previously added with **AddExamMenuItem** or **InsertAfterExamMenuItem**.

See also

AddExamMenuItem

Go to Table of Contents

2.2.88 InsertAfterExamMenuItem

This function add new menu item to the Exam menu after the specified menu item.

```
boolean InsertAfterExamMenuItem (
      BSTR Name,
      BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of new menu item, must be unique.
NameAfter	Name to insert menu item after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to insert a menu item after another previously added menu item. When one of these menu items is selected, the **EventExamMenuSelected** event will be fired.

See also

AddExamMenuItem

Go to Table of Contents

2.2.89 AddExamSubMenu

This function adds a new submenu item to the Exam menu.

```
boolean AddExamSubMenu(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add up a submenu to the Exam context menu. Items are added below it with **AddExamSubMenuItem()**.

Go to Table of Contents

2.2.90 AddExamSubMenuItem

This function adds a new submenu item to the Exam menu.

```
boolean AddExamSubMenuItem(
      BSTR Name, BSTR SubMenu);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
SubMenu	Name of submenu to which this item belongs.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenu items to the Exam context menu. A submenu must first be added with **AddExamSubMenu()**. When one of these menu items is selected, the **EventExamMenuSelected** event will be fired.

See also

AddExamSubMenu

Go to Table of Contents

2.2.91 InsertAfterExamSubMenu

This function adds a new submenu to the Exam menu, after the menu item given.

```
boolean InsertAfterExamSubMenu(
     BSTR Name, BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
NameAfter	Name of menu item this submenu should be added after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenus to the Exam context menu, after an existing item.

See also

AddExamSubMenu

Go to Table of Contents

2.2.92 InsertAfterExamSubMenuItem

This function adds a new submenu item to the Exam menu, after the submenu item given.

```
boolean InsertAfterExamSubMenuItem(
     BSTR Name, BSTR SubMenu, BSTR SubNameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
SubMenu	Name of submenu to which this item belongs.
NameAfter	Name of menu item this submenu should be added after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add a submenu item to the Exam context menu, after an existing submenu item.

See also

AddExamSubMenu

Go to Table of Contents

2.2.93 AddShelfMenuItem

This function adds a new menu item to the Shelf menu.

```
boolean AddShelfMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,111

Go to Error Codes

Remarks

Use this method to add up to five menu items to the Shelf menu. When one of these menu items is selected, the **EventShelfMenuSelected** event will be fired.

Go to Table of Contents

2.2.94 RemoveShelfMenuItem

This function removes a previously added menu or submenu from the Shelf menu.

```
boolean RemoveShelfMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110

Go to Error Codes

Remarks

Use this method to remove a menu item previously added with **AddShelfMenuItem** or **InsertAfterShelfMenuItem**.

See also

AddShelfMenuItem

Go to Table of Contents

2.2.95 InsertAfterShelfMenuItem

This function add new menu item to the Shelf menu after the specified menu item.

```
boolean InsertAfterShelfMenuItem (
      BSTR Name,
      BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of new menu item, must be unique.
NameAfter	Name to insert menu item after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to insert a menu item after another previously added menu item. When one of these menu items is selected, the **EventShelfMenuSelected** event will be fired.

See also

AddShelfMenuItem

Go to Table of Contents

2.2.96 AddShelfSubMenu

This function adds a new submenu item to the Shelf menu.

```
boolean AddShelfSubMenu(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add up a submenu to the Shelf context menu. Items are added below it with **AddShelfSubMenuItem()**.

Go to Table of Contents

2.2.97 AddShelfSubMenuItem

This function adds a new submenu item to the Shelf menu.

```
boolean AddShelfSubMenuItem(
      BSTR Name, BSTR SubMenu);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
SubMenu	Name of submenu this item belong to.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenu items to the Shelf context menu. A submenu must first be added with **AddShelfSubMenu()**. When one of these menu items is selected, the **EventShelfMenuSelected** event will be fired.

See also

AddShelfSubMenu

Go to Table of Contents

2.2.98 InsertAfterShelfSubMenu

This function adds a new submenu to the Shelf menu, after the menu item given.

```
boolean InsertAfterShelfSubMenu(
     BSTR Name, BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
NameAfter	Name of menu item this submenu should be added after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenus to the Shelf context menu, after an existing item.

See also

AddShelfSubMenu

Go to Table of Contents

2.2.99 InsertAfteShelfSubMenuItem

This function adds a new submenu item to the Shelf menu, after the submenu item given.

```
boolean InsertAfterShelfSubMenuItem(
     BSTR Name, BSTR SubMenu, BSTR SubNameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
SubMenu	Name of submenu to which this item belongs.
NameAfter	Name of menu item this submenu should be added after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add a submenu item to the Shelf context menu, after an existing submenu item.

See also

AddShelfSubMenu

Go to Table of Contents

2.2.100 AddTimelineMenuItem

This function adds a new menu item to the Timeline menu.

```
boolean AddTimelineMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add up to five menu items to the Timeline context menu. When one of these menu items is selected, the **EventMenuSelected** event will be fired.

Go to Table of Contents

2.2.101 RemoveTimelineMenuItem

This function removes a previously added menu item from the Timeline menu.

```
boolean RemoveTimelineMenuItem(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110

Go to Error Codes

Remarks

Use this method to remove a menu item previously added with **AddTimelineMenuItem** or **InsertAfterTimelineMenuItem**.

See also

AddShelfSubMenu

Go to Table of Contents

2.2.102 InsertAfterTimelineMenuItem

This function adds new menu item to the Timeline menu after the specified menu item.

```
boolean InsertAfterTimelineMenuItem (
      BSTR Name,
      BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of new menu item, must be unique
NameAfter	Name to insert menu item after

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to insert a menu item after another previously added menu item.

Go to Table of Contents

2.2.103 AddTimelineSubMenu

This function adds a new submenu item to the Timeline menu.

```
boolean AddTimelineSubMenu(
      BSTR Name);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add up a submenu to the Timeline context menu. Items are added below it with **AddTimelineSubMenuItem()**.

Go to Table of Contents

2.2.104 AddTimelineSubMenuItem

This function adds a new submenu item to the Image menu.

```
boolean AddTimelineSubMenuItem(
      BSTR Name, BSTR SubMenu);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
SubMenu	Name of submenu to which this item belong.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenu items to the Image context menu. A submenu must first be added with **AddTimelineSubMenu()**. When one of these menu items is selected, the **EventTimelineMenuSelected** event will be fired.

See also

AddTimelineSubMenu

Go to Table of Contents

2.2.105 InsertAfterTimelineSubMenu

This function adds a new submenu to the Image menu, after the menu item given.

```
boolean InsertAfterTimelineSubMenu(
      BSTR Name, BSTR NameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique
NameAfter	Name of menu item this submenu should be added after

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenus to the Image context menu, after an existing item.

See also

AddTimelineSubMenu

Go to Table of Contents

2.2.106 InsertAfterTimelineSubMenuItem

This function adds a new submenu item to the Image menu, after the submenu item given.

```
boolean InsertAfterTimelineSubMenuItem(
      BSTR Name, BSTR SubMenu, BSTR SubNameAfter);
```

Parameters

Name	Description
Name	Name of menu item, must be unique.
SubMenu	Name of submenu to which this item belongs.
NameAfter	Name of menu item this submenu should be added after.

Return Values

Value	Meaning
True	Success
False	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,10,110,111

Go to Error Codes

Remarks

Use this method to add submenu items to the Image context menu, after an existing submenu item.

See also

AddTimelineSubMenu

Go to Table of Contents

2.2.107 GetSelectedThumbnails

This function returns an XML string of all selected image windows list of ID's for all the Shelfs in a Canvas Page.

```
BSTR GetSelectedThumnails(
     BSTR CanvasPageID);
```

Parameters

Name	Description
CanvasPageID	The Canvas Page ID of the Canvas Page to retrieve selected images

Return Values

Value	Meaning
BSTR	XML string containing all selected image window in the specified Canvas Page.

Example

```
<SelectedThumbnails>
     <ThumbNail>
     <CanvasPageID>1180121602132421</CanvasPageID>
     <WindowID>918062402132425</WindowID>
     <ShelfID>524860202132421</ShelfID>
     <StudyID>1.2.124.113532.128.14747</StudyID>
     <SerieID>1.2.124.113532.128.14739</SerieID>
     <ImageID>1.2.124.113532.128.61444</ImageID>
     </ThumbNail >
</SelectedThumbnails>
```

Remarks

None.

Note: This method is available in iSite PACS versions 4.1 and higher.

See also

ListCanvasPages

Go to Table of Contents

2.2.108 GetDICOMInstanceUIDs

Returns available ImageID’s for specified SOP class within a given study.

```
BSTR GetDICOMInstanceUIDs(
     BSTR StudyID,
     BSTR SOPClassID);
```

Parameters

Name	Description
StudyID	The DICOM Study Instance UID.
SOPClassID	The DICOM SOP Class UID.

Return Values

Value	Meaning
BSTR	XML with list of available Image ID’s

Example

```
<DICOMInstanceUIDList>
     <StudyID>1.2.124.113532.128.147</StudyID>
     <SOPClassUID>1.2.124728.112852.2214619</SOPClassUID>
     <Series>
     <SerieID>16.19990728.123027.61439</SerieID>
     <SOPInstanceUID>19990728.123027</SOPInstanceUID>
     <SOPInstanceUID>119990728.123</ SOPInstanceUID>
     </Series>
</DICOMInstanceUIDList>
```

Error Codes

17, 125

Remarks

This method can only be used if the exam is currently hung in a Canvas Page.

Note: This method is available in iSite PACS versions 4.1 and higher.

Go to Table of Contents

2.2.109 SaveDICOM

DICOM File passed as buffer is sent to the iSite database for a given study.

```
long SaveDICOM(
     BSTR StudyID,
     VARIANT *DICOMData,
     Long nLength);
```

Parameters

Name	Description
StudyID	The DICOM Study Instance UID.
DICOMData	The DICOMData formatted as a SAFEARRAY.
Long	Size of DICOMData object.

Return Values

Value	Meaning
long	True if successfully sends the data to the server communication process.

Remarks

Success of the method does not indicate the data has been successfully loaded in the database. Another thread is started to complete the DICOM send and the method returns.

This method can only be used if the exam is currently hung in a Canvas Page.

Note: This method is available in iSite versions 4.1 and higher.

Go to Table of Contents

2.2.110 GetDICOMObject

Returns the DICOM data for an ImageID.

```
long GetDICOMObject(
      BSTR StudyID,
      BSTR ImageID,
      VARIANT *DICOMData);
```

Parameters

Name	**Description**
StudyID	The DICOM Study Instance UID.
ImageID	The DICOM Image Instance UID.
DICOMData	The DICOMData formatted as a SAFEARRAY.

Return Values

Value	**Meaning**
Long	Size of the DICOM data.

Error Codes

17, 125

Remarks

The function returns a 0 length buffer if an error occurs. The SAFEARRAY that’s being generated starts on index 1 for compatibility with scripting languages.

This method can only be used if the exam is currently hung in a Canvas Page.

Note: This method is available in iSite PACS versions 4.1 and higher. This method is not available for MultiFrame DICOM images.

Go to Table of Contents

2.2.111 GetLanguage

This function returns an XML stream describing the current active language selected by the user at login.

```
BSTR GetLanguage();
```

Parameters

None.

Return Values

Value	Meaning
BSTR	Returns an XML string containing active language information.

The returns an XML string with the following elements and structure.

Name	Description
Active language	Contains active language code, name, and language ID
Code	Language code
Name	Name of the language
LCID	Internal Language ID

Example

```
<Language>
     <Code>ENG</Code >
     <Name>English</Name>
     <LCID>1033</LCID>
</Language>
```

Remarks

None.

Note: This method is available in iSite PACS versions 4.1 and higher.

Go to Table of Contents

2.2.112 DeCacheImage

This method effects memory management of the iSite PACS client and is for internal use only.

```
bool DeCacheImage(
      BSTR StudyID,
      BSTR ImageID);
```

Parameters

Name	Description
StudyID	The DICOM Study Instance UID.
ImageID	The DICOM Image Instance UID.

Return Values

Value	Meaning
Bool	True if successful.

Error Codes

17, 105, 121, 125

Remarks

This method effects memory management of the iSite PACS client and is for internal use only. It is not supported for use in API integrations. If memory issues are being seen with an integration, contact APISupport@philips.com to determine if use of this method may be necessary.

Note: This method is available in iSite PACS versions 4.1 and higher.

Go to Table of Contents

2.2.113 GetListOfKeyImages

This function retrieves list of key images loaded in a Shelf.

```
BSTR GetListOfKeyImages(BSTR ShelfID);
```

Parameters

Name	Description
ShelfID	Shelf ID from which the list of Key Images is requested.

Return Values

Value	Meaning
XML	List of the Key Image UIDs
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,2,10,17,126

Go to Error Codes

Remarks

The XML String contains the following elements:

Name	Description
KeyImages	Contains the following attributes:
StudyID	The UID of the Study of the Key Image
SeriesID	The UID of the Series of the Key Image
UID	The UID of the Key Image

Example

```
<KeyImages>
       <StudyID>1.2.3.4.5.6.1</StudyID>
       <SeriesID>1.2.3.4.5.6.1</ SeriesID >
       <UID>1.2.3.4.5.6.1</UID>
</KeyImages>
```

2.2.114 GetPresentationStates

This function retrieves list of Presentation States available in the Shelf.

```
BSTR GetPresentationStates(BSTR ShelfID);
```

Parameters

Name	Description
ShelfID	Shelf ID from which the list of Presentation states is requested.

Return Values

Value	Meaning
XML	List of the Key Image UIDs
“”	Empty string, use **GetLastErrorCode()** to determine error

Error Codes

0,2,10,17,126

Go to Error Codes

Remarks

The XML String contains the following elements:

Name	Description
PresentationStates	Contains multiple presentation states.
PS	Contains the Presentation state.
PSID	The ID of the Presentation state.
Creator	The user name of the Presentation state creator.
CreationTime	The time of the Presentation state creation.
Type	The type of the Presentation state.

Example

```
<PresentationStates>
        <PS>
                <PSID>1234561</PSID>
                <Creator>John</Creator>
                <CreationTime>20:10:10</CreationTime>
                <Type>0</Type>
        </PS>
</ PresentationStates >
```

2.2.115 CopyImageDataToClipboard

This function saves the image data into a JPEG File or to the Clipboard.

```
HRESULT CopyImageDataToClipboard(BSTR ShelfID , BSTR ImageUID, VARIANT_BOOL
bAnnotations, VARIANT_BOOL bOverlays, VARIANT_BOOL bJPEGcompression);
```

Parameters

Name	Description
ShelfID	The ID of the Shelf on which the image is loaded.
ImageUID	The UID of the image which needs to be saved.
bAnnotations	True for adding the annotations on the image.
bOverlays	True for adding the overlays on the image.
bJPEGcompression	True for saving the image to a JPEG file. The file will be saved in Philips\APITemp directory with the image ID as the filename. False for saving the image to the Clipboard.

Return Values

Value	Meaning
TRUE	Success
FALSE	Failure, check for error with **GetLastErrorCode()**

Error Codes

0,2,10,17,121

Go to Error Codes

2.3 Events

2.3.1 EventExamMenuSelected

This event is fired when the user selects a menu item added with the **AddExamMenuItem** command.

Parameters

Name	Description
MenuItem	The menu item selected.
IntExamID	Internal Exam ID.

Remarks

None.

Go to Table of Contents

2.3.2 EventExamMenuSelectedEx

Similar to **EventExamMenuSelected** event, this event is also fired when the user selects a menu item added with the **AddExamMenuItem** command. However, in the second parameter, it provides an XML packet carrying all selected exams.

Parameters

Name	Description
MenuItem	The menu item selected.
IntExamIDs	An XML packet carrying all selected internal exam IDs

Example

```
<SelectedExams>
     <ExamId>11111111111111111</ExamId>
     <ExamId>22222222222222222</ExamId>
</SelectedExams>
```

Remarks

None.

Go to Table of Contents

2.4 EventMainExamShelfCreated

This event is fired only for the main shelf in the canvas page when it is created and just before it starts loading images.

Parameters

Name	Description
CanvasPageID and ShelfID	XML String containing Canvas Page ID and the Shelf ID.

Remarks

None.

Example

```
<MainExam>
            <CanvasPageID>12345678</CanvasPageID>
            <ShelfID>12345678</ShelfID>
</MainExam>
```

Go to Table of Contents

2.4.1 EventShelfMenuSelected

This event is fired when the user selects a menu item added with the **AddShelfMenuItem** command.

Parameters

Name	**Description**
MenuItem	The menu item selected.
CanvasPageID	The Canvas Page ID that owns this shelf.
ShelfID	The Shelf ID that this menu item was selected from.

Remarks

None.

Go to Table of Contents

2.4.2 EventViewMenuSelected

This event is fired when the user selects a menu item added with the **AddViewMenuItem** command.

Parameters

Name	Description
MenuItem	The menu item selected.
ContextRecord	XML string containing context information about the window.

Remarks

The context record is an XML string with the following elements and structure.

Name	Description
WindowInfo	Contains the following elements:
CanvasPageID	The Canvas Page ID for the Canvas Page that manages this window
ShelfID	The Shelf ID for the shelf that manages this window
WindowID	The Window ID of the window that fired this event
x0020000D	Study UID
x0020000E	Series UID
x00080018	SOP Instance UID

Example

```
<WindowInfo>
     <CanvasPageID>474747373</CanvasPageID>
<ShelfID>474747456</ShelfID>
     <WindowID>1904442062220000134634</WindowID>
<x0020000D>1.2.124.113532.128.147.147.112.19991023.184240.2976114</x0020000D>
<x0020000E>1.2.840.113619.2.55.1.1762384540.1757.940565163.955</x0020000E>
<x00080018>1.2.840.113619.2.55.1.1762384540.1757.940565163.956</x00080018>
</WindowInfo>
```

Go to Table of Contents

2.4.3 EventTimelineMenuSelected

This event is fired when a custom Timeline menu is selected by the user.

Parameters

Name	**Description**
bstrMenuName	Menu name of the Timeline menu selected.
bstrExamID	Internal ID of the exam that was selected in the Timeline.

Remarks

None.

Go to Table of Contents

2.4.4 EventShelfLoaded

This event is fired after a Shelf has been loaded into memory.

Parameters

Name	Description
CanvasPageID	The Canvas Page ID that owns this shelf.
ShelfID	The Shelf ID of this shelf.

Remarks

None.

Go to Table of Contents

2.4.5 EventShelfClosed

This event is fired after a Shelf exam has been closed.

Parameters

Name	Description
CanvasPageID	The Canvas Page ID that owns this shelf.
ShelfID	The Shelf ID of the closed shelf.

Remarks

None.

Go to Table of Contents

2.4.6 EventNewImagesArrived

This event is fired when new images are added to a Shelf.

Parameters

Name	Description
CanvasPageID	The Canvas Page ID that owns this shelf.
ShelfID	The Shelf ID of this shelf.
UpdatedWindowIDs	XML list of windows updated in shelf.
NewWindowIDs	XML List of new windows added to shelf.

Remarks

None.

Example

```
1234567890 0987654321
<UpdatedWindows>
    <WindowIDs>
        <WindowID>3313085152246<WindowID>
        <ImageUIDs>
                <UID>1.2.3.20047151152126.10000.20000.300000</UID>
                <UID>1.2.3.20047151152126.10000.20000.300001</UID>
        </ImageUIDs>
    </WindowIDs>
</UpdatedWindows>
<NewWindows></NewWindows>
```

Go to Table of Contents

2.4.7 EventExamMarkedRead

This event is fired when the user clicks the *MarkExamRead* button or calls iSite Radiology's **MarkExamRead()** function.

Parameters

Name	Description
ShelfID	The ShelfID that was marked read.

Remarks

This event is not fired from the **SetMarkRead()** function.

The **GetShelfStatus()** function can be used to get information such as the Internal ExamID, etc.

Go to Table of Contents

2.4.8 EventCanvasPageCreated

This event is fired when a Canvas Page is created.

Parameters

Name	Description
CanvasPageID	The Canvas Page ID for the loaded Canvas Page.

Remarks

By catching this event, you can synchronize your own custom worklist to the state of the application.

Go to Table of Contents

2.4.9 EventCanvasPageClosed

This event is fired when the user has closed a Canvas Page.

Parameters

Name	Description
CanvasPageID	The Canvas Page ID for the closed Canvas Page.

Remarks

By catching this event, you can synchronize your own custom worklist to the state of the application.

Go to Table of Contents

2.4.10 EventLogout

This event is fired when the user logs out or if the application logs out after the Idle timeout has been reached.

Parameters

Name	Description
AutoLogoutFlag	True if the logout happened to due the Idle timeout being reached.

Remarks

By catching this event, you can synchronize with the internal states of the iSite Radiology application.

Go to Table of Contents

2.4.11 EventTerminate

This event is fired when the user exits the control.

Parameters

None.

Remarks

This event is sent when the user clicks on the cancel button in the logon screen to exit the iSite Radiology application. It is responsibility of the containing application to respond to this request by exiting the application or hiding the control, etc.

Go to Table of Contents

2.4.12 EventPageStatus

This event is fired when a Folder Page visibility changes.

Parameters

Name	Description
Name	The full path name of the page that was made visible.

Type	FOLDER = Folder tab
	CANVAS = Canvas tab
	API = API Specified tab

Visible	True or False

Remarks

By catching this event, you can determine when a page is made visible or hidden. Since the Folder Page structure is hierarchical, the full path is included in the name to distinguish between pages of the same name. For example *User Folders/Interesting Cases* and *Public Folders/Interesting Cases*. The name for a Canvas Page event is the **CanvasPageID**.

Note: When a Canvas Page is created and made visible, the **EventPageStatus** event will not fire with a 'False' Visible parameter for the non-Canvas Page.

Go to Table of Contents

2.4.13 EventPreferencesApplied

This event is fired when the preferences have been written to the server.

Parameters

None.

Remarks

Use this event to check if any preferences have changed for a plug-in Preference page.

Go to Table of Contents

2.4.14 EventPreferencesApply

This event is fired when the Apply button is pressed in the Preferences dialog box, which allows validate before the preferences are written.

Parameters

Name	Description
PreferencePageName	The name of the Preference page on which the Apply button has been pressed.
PreferenceType	System, User, or Machine.

Remarks

Use this event to validate preferences and if appropriate, write preferences using **SetPreference.**

Go to Table of Contents

2.4.15 EventMediaExportStarted

This event is fired when the user clicks on the start button of the Media Export Page.

Parameters

Name	Description
DirectoryPath	Directory path which user selected for media export.

Remarks

None.

Go to Table of Contents

2.4.16 EventMediaExportCancelled

This event is fired when the user cancels the Media Export operation.

Parameters

None.

Remarks

None.

Go to Table of Contents

2.4.17 EventMediaExportComplete

This event is fired when the media export operation is complete.

Parameters

None.

Remarks

None.

Go to Table of Contents

2.4.18 EventMediaExportError

This event is fired when there is an error downloading the exams. Returns the error code and the error description.

Parameters

Name	Description
Error Code	Error returned
Error Description (XML)	XML error detail

Error Codes

209,211,212,213,214,215

Go to Error Codes

Example

The error description is in XML format. The format is:

```
<MediaExportErrorDescription>
     <ExamDetails>
            <PatientName>“Patient Name associated with the exam”</PatientName>
            <MRN>“MRN of the errored exam”</MRN>
            <Accession>“Accession number of the error exam” </Accession>
     </ExamDetails>
     <StudyDetails>
            <StudyUID>“Study UID associated with the error exam”</StudyUID>
     </StudyDetails>
     <ImageDetails>
            <ImageUID>“Image UID associated with the error exam”</ImageUID>
            <SeriesUID>“Series UID of the associated Image”</SeriesUID>
     </ImageDetails>
</MediaExportErrorDescription>
```

Remarks

The <**StudyDetails**> tag will be present if the error is a study associated error. The <**ImageDetails**> tag will be present if the error code raised is opening the image.

Go to Table of Contents

2.4.19 EventReportButtonClicked

This event is fired when the user clicks the Show Report button. This allows custom handling for buttons.

Parameters

Name	Description
bstrStudyInfo	Information generated by the exam being displayed.

Remarks

This event is only fired when clinical exam notes have been disabled and **IDXModeX** is enabled. This may be done through the ActiveX parameters **IDXModeX** and **DisableClinicalExamNotes**. This event allows you to customize the behavior at the exam level. Typical uses include loading a report into the HTML area. The **StudyInfo** is an XML string with the following elements and structure:

Name	Description
ExamInfo	Contains the following exam based elements:
IntPatientID	The internal Patient ID
IntExamID	The internal Exam ID

Example

```
<ExamInfo>
     <IntPatientID>20000</IntPatientID>
     <IntExamID>3000</IntExamID>
</ExamInfo>
```

Go to Table of Contents

2.4.20 EventShelfButton

This event is fired when the user clicks on a custom Shelf Button added through the **AddShelfButton** function.

Parameters

Name	**Description**
bstrButtonID	The Name of the shelf button clicked.
bstrShelfID	The Shelf ID containing the button clicked.

Remarks

None.

Go to Table of Contents

2.4.21 EventQueryLogout

This event is fired when the Idle timeout period has been reached – but before the application attempts to log the user out. The response to this message shall indicate to the Viewer whether the Viewer should fire the **EventLogout** message.

Parameters

Name	Description
boolAllowLogout (REF pointer)	The receiver shall set this parameter to False if they want the Viewer to close and log the user out. Set it to True and the Viewer shall stay open. Default Value: False

Remarks

None.

Go to Table of Contents

2.4.22 EventPreferenceHelp

This event is fired when the use clicks the Help button on the Preferences dialog box.

Parameters

None.

Remarks

None.

Go to Table of Contents

2.4.23 EventCacheItemAdded

This event is fired after a call to **CacheExam** but before the download begins.

Parameters

Name	Description
bstrExamID	Internal ID of the exam that was added.

Remarks

None.

Go to Table of Contents

2.4.24 EventCacheItemComplete

This event is fired when an Exam has finished caching to the Local Cache.

Parameters

Name	Description
bstrIntExamID	Internal ID of the exam that was completed
bstrIntExcpID	Internal ID of the exception that was completed.

2.4.25 EventCacheItemDeleted

This event is fired when an Exam have been deleted from the Local Cache.

Parameters

Name	Description
bstrExamID	Internal ID of the exam that was added.

Remarks

Fired in response to a call to **DeleteCachedExam**.

Go to Table of Contents

2.4.26 EventCacheItemError

This event is fired when an error occurs during a Local Cache operation.

Parameters

Name	Description
bstrExamID	Internal ID of the exam that was added.
bstrErrorMessage	Text of the error message.

Remarks

See the error message for further detail.

Go to Table of Contents

2.4.27 EventMediaExportPageClosed

This event is fired when user closes the Media Export Page.

Parameters

None.

Remarks

None.

Go to Table of Contents

2.4.28 EventAnnotationCreated

This event is fired when user creates a new Annotation or Measurement on an Image.

Parameters

Name	Description
bstrCanvasPageID	The Canvas Page ID containing the image annotated.
bstrShelfID	The Shelf ID containing the image annotated.
bstrWindowID	The Window ID of the image annotated.
bstrStudyUID	The Study UID of the image annotated.
bstrSeriesUID	The Series UID of the image annotated.
bstrImageUID	The Image UID of the image annotated.
bstrToolType	An ID that represents the type of annotation created. (see below).
bstrToken	The ID of the annotation created.

Token Value	**Meaning**
1	TOOLRULER
2	TOOLROI
3	TOOLANGLE
4	TOOLSPINELABEL
5	TOOLLINE
6	TOOLARROW
7	TOOLTRIANGLE
8	TOOLCIRCLE
9	TOOLTEXT
10	TOOLFREEHAND
11	TOOLCOMPLEXROI

Name	Description
bstrXML	XML string containing starting point XY coordinates

Remarks

The **EventAnnotationCreated** fires immediately on the mouse down event when a user clicks to create an annotation. A Create event is always followed by a **EventAnnotationModified** event when the user finishes creating the annotation and releases the mouse button. **EventAnnotationModified bstrXML** contains information specific to the type of annotation created.

bstrXML Sample

```
<AnnotationInfo>
     <StartingPoint x="338" y="207" />
</AnnotationInfo>
```

Go to Table of Contents

2.4.29 EventAnnotationModified

This event is fired when user modifies an annotation or measurement on an image.

Parameters

Name	Description
bstrCanvasPageID	The Canvas Page ID containing the image annotated.
bstrShelfID	The Shelf ID containing the image annotated.
bstrWindowID	The Window ID of the image annotated.
bstrStudyUID	The Study UID of the image annotated.
bstrSeriesUID	The Series UID of the image annotated.
bstrImageUID	The Image UID of the image annotated.
bstrToolType	An ID that represents the type of annotation created. (see below).
bstrToken	The ID of the annotation created.

Token Value	Meaning
1	TOOLRULER
2	TOOLROI
3	TOOLANGLE
4	TOOLSPINELABEL
5	TOOLLINE
6	TOOLARROW
7	TOOLTRIANGLE
8	TOOLCIRCLE
9	TOOLTEXT
10	TOOLFREEHAND
11	TOOLCOMPLEXROI

bstrXML	XML string containing information specific to the tool modified.

Remarks

bstrXML string lists information specific to the tool type.

Tool Type	bstrXML

TOOLRULER	`<AnnotationInfo>`
	`<StartingPoint x="279" y="376" />`
	`<BoundingBox left="140" top="127" right="696" bottom="735"/>`
	`<Distance>263.6</Distance>`
	`<Units>mm</Units>`
	`<Label>A</Label>`
	`<Text>A: 263.6mm</Text>`
	`<Start row="207" col="338"/>`
	`<End row="660" col="502"/>`
	`</AnnotationInfo>`

TOOLROI	`<AnnotationInfo>`
	`<StartingPoint x="363" y="466" />`
	`<BoundingBox left="108" top="169" right="471" bottom="301"/>`
	`<MeanAndSD>89.2 HU, 32 sd</MeanAndSD>`
	`<Area>6.0890 cm^2</Area>`
	`<Center row="229" col="343"/>`
	`<OuterPoint row="244" col="360"/>`
	`<Radius>14.3</Radius>`
	`</AnnotationInfo>`

TOOLANGLE	`<AnnotationInfo>`
	`<StartingPoint x="174" y="486" />`
	`<BoundingBox left="54" top="382" right="354" bottom="906"/>`
	`<Angle>95.1</Angle>`
	`</AnnotationInfo>`

TOOLSPINELABEL	`<AnnotationInfo>`
	`<StartingPoint x="313" y="80" />`
	`<BoundingBox left="1364" top="211" right="1568" bottom="415"/>`
	`<Text>C1</Text>`
	`</AnnotationInfo>`

Tool Type	bstrXML

TOOLLINE	`<AnnotationInfo>`
	`<StartingPoint x="366" y="161" />`
	`<BoundingBox left="156" top="118" right="390" bottom="320"/>`
	`<Start row="296" col="180"/>`
	`<End row="142" col="366"/>`
	`</AnnotationInfo>`

TOOLARROW	`<AnnotationInfo>`
	`<StartingPoint x="359" y="351" />`
	`<BoundingBox left="179" top="137" right="383" bottom="356"/>`
	`<Start row="161" col="203"/>`
	`<End row="332" col="359"/>`
	`</AnnotationInfo>`

TOOLTRIANGLE	`<AnnotationInfo>`
	`<StartingPoint x="260" y="170" />`
	`<BoundingBox left="54" top="120" right="284" bottom="273"/>`
	`<Vertex1 row="144" col="141"/>`
	`<Vertex2 row="249" col="78"/>`
	`<Vertex3 row="151" col="260"/>`
	`</AnnotationInfo>`

Tool Type	bstrXML

TOOLCIRCLE	`<AnnotationInfo>`
	`<StartingPoint x="399" y="268" />`
	`<BoundingBox left="326" top="189" right="406" bottom="269"/>`
	`<Center row="229" col="366"/>`
	`<OuterPoint row="249" col="399"/>`
	`<Radius>24.4</Radius>`
	`</AnnotationInfo>`

TOOLTEXT	`<AnnotationInfo>`
	`<StartingPoint x="148" y="234" />`
	`<BoundingBox left="475" top="800" right="860" bottom="1148"/>`
	`<Text>text</Text>`
	`</AnnotationInfo>`

TOOLFREEHAND	`<AnnotationInfo>`
	`<StartingPoint x="342" y="116" />`
	`<BoundingBox left="214" top="62" right="376" bottom="184"/>`
	`<Points count="20"/>`
	`</AnnotationInfo>`

TOOLCOMPLEXROI	`<AnnotationInfo>`
	`<StartingPoint x="335" y="117" />`
	`<BoundingBox left="236" top="-158" right="962" bottom="512"/>`
	`<Points count="49"/>`
	`<MeanAndSD>89.2 HU, 32 sd</MeanAndSD>`
	`<Area>5805.5</Area>`
	`<CenterOfMass row="185" col="602"/>`
	`<Text>58.055 cm^2</Text>`
	`</AnnotationInfo>`

Go to Table of Contents

2.4.30 EventAnnotationDeleted

This event is fired when user deletes an Annotation or Measurement on an Image.

Parameters

Name	Description
bstrCanvasPageID	The Canvas Page ID containing the image annotated.
bstrShelfID	The Shelf ID containing the image annotated.
bstrWindowID	The Window ID of the image annotated.
bstrStudyUID	The Study UID of the image annotated.
bstrSeriesUID	The Series UID of the image annotated.
bstrImageUID	The Image UID of the image annotated.
bstrToolType	An ID that represents the type of annotation created. (see below).
bstrToken	The ID of the annotation created.

Token Value	Meaning
1	TOOLRULER
2	TOOLROI
3	TOOLANGLE
4	TOOLSPINELABEL
5	TOOLLINE
6	TOOLARROW
7	TOOLTRIANGLE
8	TOOLCIRCLE
9	TOOLTEXT
10	TOOLFREEHAND
11	TOOLCOMPLEXROI

Name	Description
bstrXML	"<AnnotationInfo> </AnnotationInfo>"

Remarks

None.

Go to Table of Contents

2.4.31 EventPresentationStateSaved

This event is fired when a Presentation State is saved either through the **SavePresentationState** method or though the GUI.

Parameters

Name	**Description**
bstrCanvasPageID	Canvas Page ID the Presentation State was saved from.
bstrShelfID	Shelf ID the Presentation State was saved from.
bstrPSName	Unique Presentation State ID.

Remarks

None.

Go to Table of Contents

2.4.32 EventPresentationStateLoaded

This event is fired when a Presentation State is loaded into a shelf either through the **LoadPresentationState** method or though the GUI.

Parameters

Name	Description
bstrCanvasPageID	Canvas Page ID the Presentation State was saved from.
bstrShelfID	Shelf ID the Presentation State was saved from.
bstrPSName	Unique Presentation State ID.

Remarks

None.

Go to Table of Contents

2.4.33 EventImageWindowCreated

This event is fired once for each thumbnail, popup, and diagnostic window as they are created.

Parameters

Name	Description
bstrCanvasPageID	Canvas Page ID for the image displayed.
bstrShelfID	Shelf ID for the image displayed.
bstrWindowID	Window ID for the image displayed.
bstrStudyUID	Study UID for the image displayed.
bstrSeriesUID	Series UID for the image displayed.
bstrImageUID	Image UID for the image displayed.
nWindowType	

Long	**Description**
0	Thumbnail
1	Popup
2	Diagnostic

Remarks

None.

Go to Table of Contents

2.4.34 EventMenuOpening

This event is fired whenever the user right clicks to show a context menu.

Parameters

Name	Description	
nMenuType	**Long**	**Description**
	1	Exam menu
	2	Shelf menu
	3	View menu
	4	Timeline menu
oMenu	The **iSiteContextMenu** object representing the menu which is about to be shown	

Remarks

None.

Go to Table of Contents

3 ISiteContextMenu Object

The **iSiteContextMenu** object allows integrators to modify the contents of the context menus used in the iSite PACS clients. When a user right clicks on a window, the client will fire an EventMenuOpening event passing the menu type and a pointer to a copy of this object.

Note: The **iSiteContextMenu** Object is not compatible with Scripting languages such as Javascript and VBScript. For context menu operations via script, use the traditional iSite PACS API context menu functions such as AddViewMenuItem, AddExamMenuItem, AddShelfMenuItem, AddTimelineMenuItem, etc.

3.1 Methods

3.1.1 AddSeparator

This method places a menu separator after the last item on the menu.

```
Boolean AddSeparator();
```

Return Values

Value	Meaning
True	The method completed successfully.

Remarks

None.

Go to Table of Contents

3.1.2 InsertSeparator

This method inserts a menu separator immediately after the specified menu item.

```
Boolean InsertSeparator(LONG nIDorPos, LONG nFlag);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID. **Value** / **Description** 0 / nIDorPos is an ID (MF_BYCOMMAND) 400 / nIDorPos is a position (MF_BYPOSITION)

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item was not found.

Remarks

Specifying a position of -1 will insert the separator before the first item.

Go to Table of Contents

3.1.3 AddMenuItem

This method appends a new menu item to the end of the menu.

```
Boolean AddMenuItem(BSTR bstrText, LONG *pnID);
```

Parameters

Name	Description
bstrText	The text that will be displayed on the menu for this item.
pnID	Pointer to a variable to set to the ID of the new item.

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified text matches an existing menu item.

Remarks

None.

Go to Table of Contents

3.1.4 AddKnownMenuItem

This method appends a new menu item to the end of the menu and assigns the specified ID.

```
Boolean AddKnownMenuItem(LONG nID, BSTR bstrText);
```

Parameters

Name	Description
nID	Menu item ID for this item.
bstrText	The text that will be displayed on the menu for this item.

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified text matches an existing menu item.

Remarks

This method should **not** be used by integrators as it will cause unpredictable results.

Go to Table of Contents

3.1.5 InsertNewMenuItem

This method inserts a menu item immediately after the specified menu item and assigns an ID for the new menu item.

```
Boolean InsertNewMenuItem(LONG nIDorPos, LONG nFlag, BSTR bstrText, LONG* pnID);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the nIDorPos is a position or an ID.

Value	Description
0	nIDorPos is an ID (MF_BYCOMMAND)
400	nIDorPos is a position (MF_BYPOSITION)

Name	Description
bstrText	The text that will be displayed on the menu for this item.
pnID	Pointer to a variable to set to the ID of the new item.

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn't exist or the specified text matches an existing menu item.

Remarks

Specifying a position of -1 will insert the new menu item before the first item.

Go to Table of Contents

3.1.6 InsertKnownMenuItem

This method inserts a menu item immediately after the specified menu item and assigns the specified ID for the new menu item.

```
Boolean InsertKnownMenuItem(LONG nIDorPos, LONG nFlag, LONG nID, BSTR bstrText);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID.

Value	Description
0	nIDorPos is an ID (MF_BYCOMMAND)
400	nIDorPos is a position (MF_BYPOSITION)

Name	Description
nID	The ID of the new item.
bstrText	The text that will be displayed on the menu for this item.

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn’t exist or the specified text matches an existing menu item.

Remarks

This method should **not** be used by integrators as it will cause unpredictable results. Specifying a position of -1 will insert the new menu item before the first item.

Go to Table of Contents

3.1.7 AddSubmenu

This method creates a new submenu and appends it to the end of menu.

```
Boolean AddSubmenu(BSTR bstrText, IDispatch** ppSubmenu);
```

Parameters

Name	Description
bstrText	The text that will be displayed on the menu for this item.
pSubmenu	A pointer to the newly created submenu

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified text matches an existing menu item or the submenu could not be created because of an internal error.

Remarks

Internal errors include such things as running out of virtual memory. If the method fails, the submenu point is set to NULL. The calling routine is responsible for releasing the returned object.

Go to Table of Contents

3.1.8 GetSubmenu

This method returns the submenu located at the specified position.

```
Boolean GetSubmenu(LONG nPos, IDispatch** ppSubmenu);
```

Parameters

Name	Description
nPos	The zero-based position or the ID of the desired submenu.
pSubmenu	A pointer to the requested submenu

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified position is not valid or the item at the specified position is not a submenu.

Remarks

A position is required rather than and ID because submenus aren't assigned IDs when added or inserted. If the method fails, the submenu point is set to NULL. The calling routine is responsible for releasing the returned object.

Go to Table of Contents

3.1.9 InsertSubmenu

This method created a new submenu and inserts it immediately after the menu item specified by **nIDorPos**. The new submenu is returned.

```
Boolean InsertSubmenu(LONG nIDorPos, LONG nFlag, BSTR bstrText, IDispatch**
pSubmenu);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID. **Value** / **Description** 0 / nIDorPos is an ID (MF_BYCOMMAND) 400 / nIDorPos is a position (MF_BYPOSITION)
bstrText	The text that will be displayed on the menu for this item.
pSubmenu	A pointer to the newly created submenu

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn’t exist or the specified text matches an existing menu item or the submenu could not be created because of an internal error.

Remarks

Specifying a position of -1 will insert the new submenu before the first item. Internal errors include such things as running out of virtual memory. The calling routine is responsible for releasing the returned object.

Go to Table of Contents

3.1.10 GetText

This method returns the text for an existing menu item.

```
Boolean GetText(LONG nIDorPos, LONG nFlag, BSTR* pbstrText);
```

Parameters

Name	Description
`nIDorPos`	Either a zero-based position or the ID of a menu item.
`nFlag`	Determines whether the **nIDorPos** is a position or an ID. <table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>nIDorPos is an ID (MF_BYCOMMAND)</td></tr><tr><td>400</td><td>nIDorPos is a position (MF_BYPOSITION)</td></tr></table>
`pbstrText`	A pointer to the text that is displayed on the menu for this item.

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn’t exist.

Remarks

If the method fails the value of the text is set to NULL. **pbstrText** is assumed to point to an uninitialized BSTR and no attempt is made to release the string before execution. This could lead to a memory leak if the calling routine doesn’t free the old string. The calling routine is responsible for freeing the returned string. There is no text associated with a separator, so the method will return NULL if **nIDorPos** specifies one.

Go to Table of Contents

3.1.11 SetText

This method replaces the existing text for a menu item with the new text.

```
Boolean SetText(LONG nIDorPos, LONG nFlag, BSTR bstrText);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID.

Value	Description
0	nIDorPos is an ID (MF_BYCOMMAND)
400	nIDorPos is a position (MF_BYPOSITION)

Name	Description
bstrText	The new text that will be displayed on the menu for this item.

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn't exist or the specified text matches an existing menu item.

Remarks

None.

Go to Table of Contents

3.1.12 Remove

This method removes the specified item from the menu. It also deletes the text and, if the item is a submenu, releases the submenu object.

```
Boolean Remove(LONG nIDorPos, LONG nFlag);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID.

Value	Description
0	nIDorPos is an ID (MF_BYCOMMAND)
400	nIDorPos is a position (MF_BYPOSITION)

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn’t exist.

Remarks

None.

Go to Table of Contents

3.1.13 Clear

This method removes all items from the menu. It also deletes the text and releases any submenu objects.

```
Boolean Clear();
```

Return Values

Value	Meaning
True	The method completed successfully.

Remarks

This method should **not** be called by the integrator as it will interfere with normal operations.

Go to Table of Contents

3.1.14 CheckMenuItem

This method adds or removes a check mark from next to the specified menu item.

```
Boolean CheckMenuItem(LONG nPosOrID, LONG nType, VARIANT_BOOL bCheck);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the nIDorPos is a position or an ID.

Value	Description
0	nIDorPos is an ID (MF_BYCOMMAND)
400	nIDorPos is a position (MF_BYPOSITION)

bCheck	Indicates whether the item should be checked (True) or unchecked (False).

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn't exist or is not a menu item.

Remarks

The method doesn't allow separators or submenus to be checked.

Go to Table of Contents

3.1.15 IsChecked

This method is used to determine if the specified item has a check mark next to it.

```
Boolean IsChecked(LONG nIDorPos, LONG nType);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID.

Value	Description
0	nIDorPos is an ID (MF_BYCOMMAND)
400	nIDorPos is a position (MF_BYPOSITION)

Return Values

Value	Meaning
True	The specified menu item exists, is a menu item and is checked.
False	The specified menu item doesn't exist or is not a menu item or is not checked.

Remarks

None.

Go to Table of Contents

3.1.16 EnableMenuItem

This method will enable or disable the specified menu item. Disabled items will be grayed out and not selectable by the user.

```
Boolean EnableMenuItem(LONG nIDorPos, LONG nFlags, VARIANT_BOOL bEnable);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID. **Value** — **Description** 0 — nIDorPos is an ID (MF_BYCOMMAND) 400 — nIDorPos is a position (MF_BYPOSITION)
bCheck	Indicates whether the item should be enabled (True) or disabled (False).

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified menu item doesn’t exist or is not a menu item.

Remarks

The method doesn’t allow separators or submenus to be disabled.

Go to Table of Contents

3.1.17 GetCount

This method returns the number of items in the menu.

```
  Boolean GetCount(LONG* pnCount);
```

Parameters

Name	Description
pnCount	A pointer to a LONG that will receive the number of items.

Return Values

Value	Meaning
True	The method completed successfully.

Remarks

None.

Go to Table of Contents

3.1.18 Load

This method will load the specified menu from the specified file.

```
Boolean Load(BSTR bstrFileName, LONG nID);
```

Parameters

Name	**Description**
bstrFileName	The name of the file containing the menu to be loaded.
nID	The ID of the menu resource to be loaded.

Return Values

Value	**Meaning**
True	The method completed successfully.
False	The specified file doesn't exist or doesn't contain the specified menu resource or the menu couldn't be loaded for an internal reason.

Remarks

This function should **not** be called by the integrator as it will clear the existing menu and replace it with the new one. Use of this function could cause unpredictable results and disable come client functionality.

Go to Table of Contents

3.1.19 CopyToMenu

This method copies the menu to the specified external Windows menu.

```
Boolean CopyToMenu(LONGLONG nhMenu);
```

Parameters

Name	Description
nhMenu	The HMENU representing the Window menu to receive a copy of the menu.

Return Values

Value	Meaning
True	The method completed successfully.
False	The member failed for an internal reason.

Remarks

This call is recursive, copying all submenus as well. It is not intended for use by the integrator.

Go to Table of Contents

3.1.20 GetRange

This method retrieves the range of IDs from which the menu will assign new IDs.

```
Boolean GetRange(LONG* pnMin, LONG* pnMax);
```

Parameters

Name	Description
pnMin	Pointer to a LONG that will receive the minimum ID that can be assigned.
pnMax	Pointer to a LONG that will receive the maximum ID that can be assigned.

Return Values

Value	Meaning
True	The method completed successfully.

Remarks

None.

Go to Table of Contents

3.1.21 SetRange

This method sets the range from which the menu will assign new IDs.

```
  Boolean SetRange(LONG nMin, LONG nMax);
```

Parameters

Name	Description
nMin	The minimum ID that can be assigned.
nMax	The maximum ID that can be assigned.

Return Values

Value	Meaning
True	The method completed successfully.
False	nMin > nMax or a menu item has already been added to the menu.

Remarks

This method can only be called before the first menu item is added.

Go to Table of Contents

3.1.22 Popup

This method displays the menu at the specified screen coordinates.

```
Boolean Popup(LONGLONG nWnd, LONG nX, LONG nY);
```

Parameters

Name	Description
nWnd	The HWND of the Window to receive any message caused by the user clicking on a menu item.
nX	The X screen coordinate of the location to display the menu.
nY	The Y screen coordinate of the location to display the menu.

Return Values

Value	Meaning
True	The method completed successfully.
False	The method failed for an internal reason.

Remarks

None.

Go to Table of Contents

3.1.23 Find

This method scans the menu looking for an item with the specified text.

```
Boolean Find(BSTR bstrText, LONG* pnPos);
```

Parameters

Name	Description
bstrText	The text to search for. It must exactly match the menu item text.
pnPos	Pointer to the LONG that will receive the position of the menu item.

Return Values

Value	Meaning
True	The specified text was found.
False	The specified text was not found.

Remarks

This method is not recursive. The integrator is responsible for traversing the submenus if that is the desired behavior. If the text is not found, the position is set to -1.

Go to Table of Contents

3.1.24 GetType

This method returns a value that indicates whether the specified item is a menu item, a submenu or a separator.

```
Boolean GetType(LONG nIDorPos, LONG nFlag, LONG* pnType);
```

Parameters

Name	Description
nIDorPos	Either a zero-based position or the ID of a menu item.
nFlag	Determines whether the **nIDorPos** is a position or an ID. **Value** — **Description** 0 — nIDorPos is an ID (MF_BYCOMMAND) 400 — nIDorPos is a position (MF_BYPOSITION)
pnType	Pointer to the LONG that will receive the type of the menu item. **Value** — **Description** 0 — The item is a menu item. 1 — The item is a submenu. 2 — The item is a separator.

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified item doesn't exist.

Remarks

If the item doesn't exist, the type is set to -1.

Go to Table of Contents

3.1.25 SetBitmaps

This method sets two bitmaps to be used to check the specified menu item overriding the Windows default check mark.

```
Boolean SetBitmaps(LONG nIDorPos, LONG nFlag, LONGLONG nhUncheckedBMP, LONGLONG
nhCheckedBMP);
```

Parameters

Name	Description		
nIDorPos	Either a zero-based position or the ID of a menu item.		
nFlag	Determines whether the **nIDorPos** is a position or an ID.		
		Value	**Description**
		0	nIDorPos is an ID (MF_BYCOMMAND)
		400	nIDorPos is a position (MF_BYPOSITION)
nhUncheckedBMP	The HBITMAP of the unchecked bitmap.		
nhCheckedBMP	The HBITMAP of the checked bitmap.		

Return Values

Value	Meaning
True	The method completed successfully.
False	The specified item doesn't exist.

Remarks

If both bitmaps are the same the image will always show. The bitmaps should be 13x13 bits in area.

Go to Table of Contents