Kodak

Smart Review System 1.0

API Reference Manual

Document Number
719-00920A-EN-RevA

19 December 2007

Copyright 2007 Kodak. All rights reserved.

Kodak

Smart Review System 1.0
API Reference Manual

CONTENTS

1. Overview 1

2. SRS Service Architecture 1

3. API Method Reference 3
   - 3.1 Session Management 3
     - 3.1.1 Overview 3
     - 3.1.2 CreateSession 4
     - 3.1.3 CreateSession2 6
     - 3.1.4 CreateMultiDocumentSession 8
     - 3.1.5 DestroySession 11
     - 3.1.6 EnumerateOpenSessions 12
     - 3.1.7 OnSessionClosed (Callback) 13
   - 3.2 Annotations 13
     - 3.2.1 Overview 13
     - 3.2.2 GetNotes2 15
     - 3.2.3 DeleteNotes 16
     - 3.2.4 OnSynchronizeNotes (Callback) 17
   - 3.3 Approval 19
     - 3.3.1 Overview 19
     - 3.3.2 GetApprovalStatusList 20
     - 3.3.3 SetApprovalStatusList 22
     - 3.3.4 GetDefaultApprovalStatusList 23
     - 3.3.5 SetDefaultApprovalStatusList 23
     - 3.3.6 GetUserApprovalStatus 24
     - 3.3.7 SetUserApprovalStatus 25
     - 3.3.8 GetUsersApprovalStatus 26
     - 3.3.9 GetDocumentApprovalStatus 27
     - 3.3.10 SetDocumentApprovalStatus 27
     - 3.3.11 SetApproveOnClose 28
     - 3.3.12 OnUserApproval (Callback) 29
     - 3.3.13 SynchronizeUserApproval (Callback) 30
     - 3.3.14 UserLogin (Callback) 31
     - 3.3.15 UserLogout (Callback) 32
   - 3.4 User Roles and Permissions 32
     - 3.4.1 Overview 32
     - 3.4.2 GetUserRole 34
     - 3.4.3 SetUserRole 34
     - 3.4.4 RemoveUserRole 35
     - 3.4.5 AdjustUserPermissions 35
     - 3.4.6 QueryUserPermissions 36
   - 3.5 Color 36

3.5.1 Overview 36
3.5.2 GetColorParameters2 37
3.5.3 GetDefaultColorParameters2 38
3.5.4 SetColorParameters2 38
3.5.5 SetDefaultColorParameters2 39
3.6 File Management and Metadata 40
3.6.1 Overview 40
3.6.2 Get File Info 40
3.6.3 GetFileType 41
3.6.4 GetThumbnail 42
3.6.5 ObsoleteSource 43
3.6.6 GetMetaData 43
3.6.7 SetMetaData 44
3.7 Server Utilities 46
3.7.1 GetVersionInfo 46
3.7.2 GetServerInfo 46
3.7.3 RestartServer 47
3.8 Data Dictionary 48
3.8.1 Simple Datatypes 48
3.8.2 Complex Datatypes 49
3.8.3 Exceptions 55

4. Appendix A: Metadata Structure 56
4.1 Page Annotations 57
4.2 Approval Status List 65
4.3 SRS Server Version 67
4.4 Color Parameter 68
4.5 Document Approval 69
4.6 Users Approval Status 71

1. OVERVIEW

This document provides reference information for the Application Programming Interface (API) for the Kodak® Smart Review System (SRS) 1.0 software.

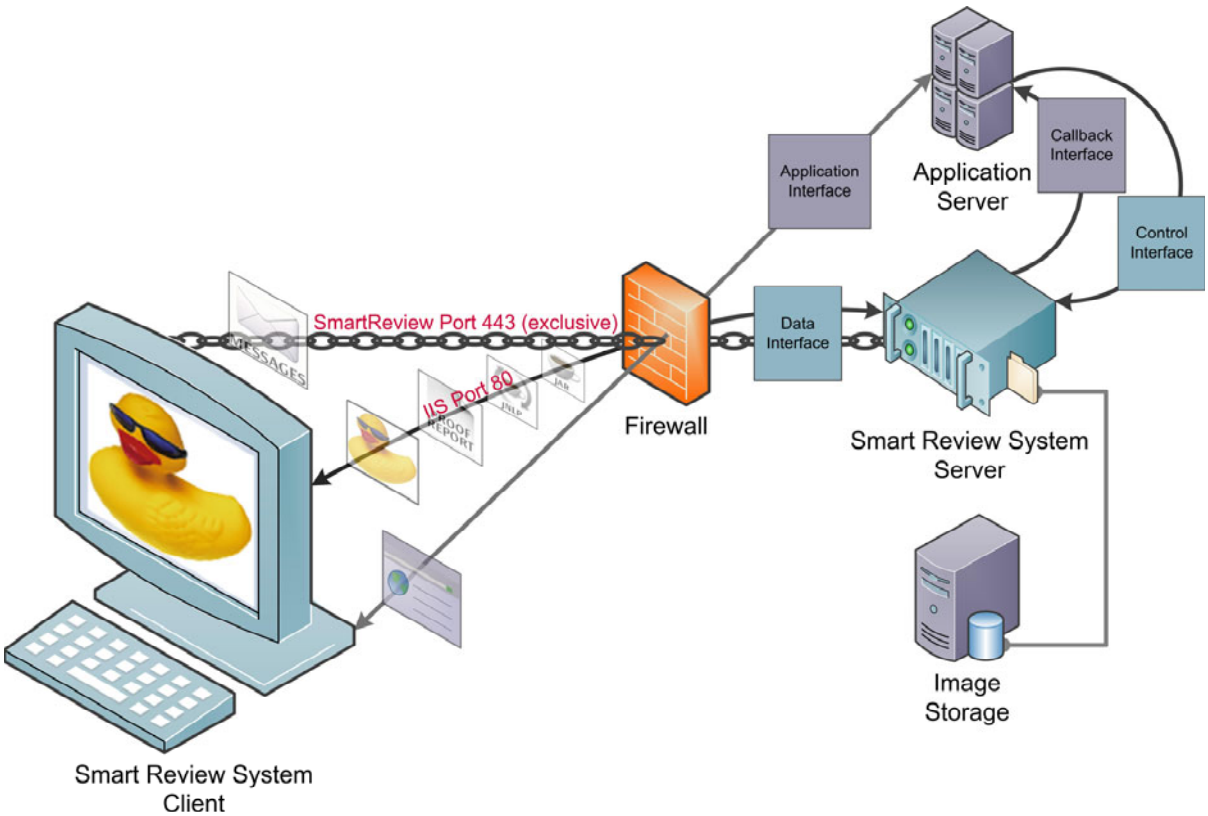
The Web Services Description Language (WSDL) file for this interface can be queried from a working SRS server at **http://<serverurl>/proofpartner/PartnerAPI.asmx?WSDL** (for example, http://64.165.113.138/proofpartner/PartnerAPI.asmx?WSDL). Both the PartnerAPI.wsdl and PartnerAPICallback.wsdl files are available on the SRS DVD.

Note: *Italics* are used throughout the text to highlight differences in functionality between SRS and the Kodak RealTimeProof™ (RTP) Partner software.

This document also refers to configuration settings in Smart Review System Administration. To open Smart Review System Administration on the SRS server, select **Start > Programs > Kodak SRS > SRS Administrator**, or click the **SRS Administrator** shortcut on the desktop. Alternatively, you can access the administration Web pages remotely by typing the URL **http://<serverurl>:6377/TwAmWebAdmin** in a Web browser. *Smart Review System Administration is the replacement for the RTP Partner.conf file.*

Note: Remote access to the URL requires Administrator credentials. We do not recommend that you use the PARTNER account credentials for anything but the SRS services.

2. SRS SERVICE ARCHITECTURE



1. A client computer uses a Web browser to access a launch link provided on the Application Interface.
2. When the Application Server detects that there is an image to be opened by the Smart Review System Server, it calls the Control Interface of the service to initiate a session and provides it the location information of the original file on the Image Storage.
3. The SRS Server accesses this image file and opens a session.
4. The Control Interface returns a reference to the session, and the Application Server sends a response to the Web browser with this session reference.
5. The Application Interface can either:
    - Use a system call that launches the Installed SRS client executable on the client computer with session reference URL as an argument. The installed SRS client executable will open the session reference URL (over port 80).
    - Redirect the client computer's Web browser to the session reference URL to launch the JNLP SRS client. On the first time a client machine connects to the SRS Server and on subsequent upgrades of the SRS Server, there will be a download of JAR files (over port 80). On every session launch, there will be a download of a small JNLP file (over port 80).
6. On reaching the session reference, processing is triggered on the SRS Server, and an image renders on the SRS client.
    - All messages between the SRS client and SRS Server occur across the Data Interface (over port 443). This message traffic includes approvals and annotation data and is secure. In order for SRS to work properly, the SRS Server must have exclusive use of this port. If the Server cannot lock on this port, the Server will not be able to communicate the original image to the SRS client. This messaging port can be configured through **SRS Administration > Security > Communication Port**.
    - The SRS client fetches image file data, cache data, proof reports, and TIFs printed to file from the SRS Server across the Data Interface (over port 80).
7. SRS Server sends the client computer's activity messages to the Callback Interface published by the Application Server.

Note: All references to port 80 assume that SSL is not configured. If SSL is configured then these ports are 443, but some speed will be compromised. More information about configuring your SRS server for SSL is available on request.

3. API METHOD REFERENCE

The API Method reference is broken down into functional groups as follows:

- Session management
- Annotations
- Approval
- User roles and permissions
- Color
- File management and metadata
- Server utilities

Each functional group has a corresponding section in this document. Each section begins with an overview and explanation of key concepts and terms relevant to that functional group.

The full data dictionary (details of fields in API data structures) is provided at the end of this document.

3.1 Session Management

3.1.1 Overview

Client Launch

The client launch procedure differs slightly between RTP Partner and SRS.

In RTP Partner, the client is launched by delivering a bootstrap file, with extension ppc or ppp, which contains a session reference returned from a session creation API.

In SRS, the client may be launched in one of two ways. The first is by installing the client on the computer, and then simply prepending **smartreview**: to the launchURL (session reference) returned from a session creation API. The resultant command is typically executed directly in order to invoke the client.

The alternative approach to launching the client is via Java Network Launch Protocol (JNLP). To launch using this method, Java must be installed on the client computer, and there is no need to prepend **smartreview**: to the launchURL. In other words, the URL may be used directly.

Document Roots

All source files used in the SRS APIs must be placed in a location known to SRS. These locations are referred to as Document Roots, or DocRoots, and are configured in Smart Review System Administration under **Document Roots**.

The system allows one **Default DocRoot** and multiple alternative **DocRoot Directories**. Alternative DocRoots are known to the SRS system by their configured names. (For more information, see the *Smart Review System 1.0 Administration Guide*.)

The SRS APIs expect that the Source input parameters specify files relative to DocRoots. For example:

- If the file **FileA.pdf** is located in **FolderA** in the Default DocRoot, the Source input parameter to an SRS API is `FolderA\FileA.pdf`.
- If the file **FileB.pdf** is located in **FolderB** in the first alternative DocRoot (named DocRoot1), the Source input parameter to an SRS API is `DocRoot1\FolderB\FileB.pdf`.

A session's lifetime is from its creation until it is explicitly destroyed by exiting the client, the DestroySession method, or by automatic expiration after a configurable period of inactivity (**Session > Session Timeout**).

Notes

The session IDs returned from the session APIs allow for full or partial URLs. You can configure the computer name or IP address in the returned URL under **Session > Server Name** and the full or partial URL under **Session > Generate Full URL**.

3.1.2 CreateSession

The CreateSession method is called to open an imaging session for a source image file. The interface returns a session that is valid for a single client only. This method also adds the file into SRS (unless it has already been added by a previous call), creates database entries, and performs image pre-caching as specified in the render cache settings (**Rendering**).

Input Parameter	Type	Description
HostSessionID	String	A session identification string used by the host application. The service includes this ID in the callbacks.
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or to one of the **Document Roots > Alternatives**.
SessionType	String	Any configured user role. The default configuration includes the Viewer and Reviewer roles. The **Authorize On Approval** option may optionally be specified by adding the string "`,Approval/Authorize= on`" or "`,Approval/Authorize=off`" to the SessionType parameter. For more information see section 3.3 Approval. The **OnApproval** option may optionally be specified by adding the string "`,OnApproval=<user role name>`" to the SessionType parameter. The value of <user role name> may be any configured user role. This option will apply the specified user role to the session after an approval is set for this session via the SRS client.

Input Parameter	Type	Description
UserID	String	Represents a unique user identification in the host application. Used as user identification in annotations and approval. The UserID cannot contain spaces.
FirstName	String	Human-readable user name. Used to display the user name in the Smart Review client.
LastName	String	
ClientIP	String	(optional) IP address that is allowed to access this session.
Secure	Boolean	This parameter should be **True** if SRS is configured to use SSL and **False** otherwise. This flag controls whether the returned Session Reference begins with HTTPS (Secure=True) or HTTP (Secure=False).
StreamingMode	StreamType	*This parameter is not used in SRS but is required to maintain backwards compatibility with RTP Partner. Because this parameter is an Enumeration type, a valid value must be passed (**RTI** or **JPEG**).*
CallbackURL	String	(optional) URL that RTP Partner uses to call callback methods following annotation or approval events in this session. The default value is specified in **Callback > Callback URL**.

Output Parameter	Type	Description
SessionReference	String	A URL identifying the session—for example: http://partnertest/TWAMWeb/Views/SpeDetailsView/SmartReview.jnlp?SessionId=CB9BA1FA8520004B9F0C719B9B149917&CustId=6BB67BC62F3A02A32F3A02A32F3A02A4&ContainerId=6A57DBC62F3A02A32F3A02A32F3A02A4&SpeId=AAECC60B4DCFEEDF0DB83DCEC1326DB9&paramFile=CB9BA1FA8520004B9F0C719B9B149917&SessId=CB9BA1FA8520004B9F0C719B9B149917&jobId=6A57DBC62F3A02A32F3A02A32F3A02A4 In this example, "`partnertest`" is the SRS server name. This portion of the returned URL is configurable at **Configuration Settings > Smart Review System Settings > Server URL**. If **Configuration Settings > Smart Review System Settings > GenerateFullURL** is not selected, the SessionReference will not contain `http://partnertest`.

Fault Codes		Description
		Client.FileNotFound

		Client.SyntaxError Server.InternalError Server.UnsupportedFileFormat Client.UnsupportedParameter Client.AccessForbidden

3.1.3 CreateSession2

The CreateSession2 method is called to open an imaging session for a source image file. The interface returns a session that is valid for a single client only. This method also adds the file into the SRS system (unless it has already been added by a previous call), creates database entries, and performs image pre-caching as specified in the render cache settings (**Rendering**).

Input Parameter	Type	Description
HostSessionID	String	A session identification string used by the host application. The service includes this ID in the callbacks.
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
Platform	PlatformType	*This parameter is not used in SRS but is required to maintain backwards compatibility with RTP Partner.* *Because this parameter is an Enumeration type, a valid value must be passed.* Enumerated values are: • **WinNT** for Windows NT/2000 and later • **MacOS** for Apple® Macintosh® • **Win9x** for Windows 95 and Windows 98
SessionType	String	Any configured user role. The default configuration includes the Viewer and Reviewer roles. The Authorize On Approval option may optionally be specified by adding the string "`,Approval/Authorize= on`" or "`,Approval/Authorize=off`" to the SessionType parameter. For more information see section 3.3 Approval.

Input Parameter	Type	Description
		The **OnApproval** option may optionally be specified by adding the string ",OnApproval=<user role name>" to the SessionType parameter. The value of <user role name> may be any configured user role. This option will apply the specified user role to the session after an approval is set for this session via the SRS client.
UserID	String	Represents a unique user identification in the host application. Used as user identification in annotations. The UserID cannot contain spaces.
FirstName	String	Human-readable user name. Used to display user name in the Smart Review client.
LastName	String	
ClientIP	String	(optional) IP address that is allowed to access this session.
Secure	Boolean	This parameter should be **True** if SRS is configured to use SSL and **False** otherwise. This flag controls whether the returned Session Reference begins with HTTPS (Secure=True) or HTTP (Secure=False).
StreamingMode	StreamType	*This parameter is not used in SRS but is required to maintain backwards compatibility with RTP Partner. Because this parameter is an Enumeration type, a valid value must be passed (**RTI** or **JPEG**).*
CallbackURL	String	(optional) URL that RTP Partner uses to call callback methods following annotation or approval events in this session. The default value is specified in **Callback > Callback URL**.

Output Parameter	Type	Description
Launch	LaunchParams	Contains the following element: **Viewer**: LaunchParamsViewer. Viewer.URL is a URL identifying the session. For more information, see the description of the SessionReference output parameter in section 3.1.2 CreateSession. *In SRS, the Viewer.Version (required client*

		version) and Viewer.VersionURL (URL to download the client) are set to the string "ParameterNotSupportedInSRS". These parameters are not used in SRS but are returned to maintain backwards compatibility with RTP Partner.

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Server.UnsupportedFileFormat Client.UnsupportedParameter Client.AccessForbidden

3.1.4 CreateMultiDocumentSession

The CreateMultiDocumentSession method is called to launch an imaging session with multiple source image files. The CreateMultiDocumentSession method can also be used to add more source images to an existing MultiDocumentSession by passing in a session URL returned from a prior invocation of CreateMultiDocumentSession as the BatchID. If the session is open in the Smart Review client at the time of the second CreateMultiDocumentSession call, the client is updated with the additional source images. When adding documents to existing session, the host application does not send a URL to the user as it does in the usual launch sequence.

Input Parameter	Type	Description
HostSessionID	String	A session identification string used by the host application. The service includes this ID in the callbacks.
Source	SourceDescription[ ]	Each SourceDescription object contains the following fields: • **Source**: Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.. • **Name**: file name to be displayed in the RTP client. Must be set if imaging file storage name does not match its logical name in the host application; otherwise it will be calculated based on the Source member

Input Parameter	Type	Description
		provided. *This parameter is not used in SRS.* • **HierarchyPath**: imaging file hierarchy path. Must be set if the source URL is different than the image's logical name in the host application. *This parameter is not used in SRS.* • **Version**: imaging source file version. Default is **1**. *This parameter is not used in SRS.* • **FileID**: unique string key identifying the image file. FileID can be identical (not empty) for more than one image file, if these are different versions of the same image file; in this case the client will view these images in Compare mode. *This parameter is not used in SRS.* • **CurrentPage**: page to be displayed during first view. Default is **1**. *This parameter is not used in SRS.*
Platform	PlatformType	*This parameter is not used in SRS but is required to maintain backwards compatibility with RTP Partner.* *Because this parameter is an Enumeration type, a valid value must be passed.* Enumerated values are: • **WinNT** for Windows NT/2000 and later • **MacOS** for Mac • **Win9x** for Windows 95 and Windows 98
SessionType	String	Any configured user role. The default configuration includes the Viewer and Reviewer roles. The **Authorize On Approval** option may optionally be specified by adding the string "`,Approval/Authorize= on`" or "`,Approval/Authorize=off`" to the SessionType parameter. For more information see section 3.3 Approval.

Input Parameter	Type	Description
UserID	String	Represents a unique user identification in the host application. Used as user identification in annotations. The UserID cannot contain spaces.
FirstName	String	Human-readable user name. Used to display the user name in the Smart Review client.
LastName	String	
ClientIP	String	(optional) IP address that is allowed to access this session.
Secure	Boolean	This parameter should be **True** if SRS is configured to use SSL and **False** otherwise. This flag controls whether the returned Session Reference begins with HTTPS (Secure=True) or HTTP (Secure=False).
NumInitialImages	Integer	*This parameter is not used in SRS; all images in the multi-document session are available in the client.*
BatchID	String	(optional) The BatchID returned from a previous CreateMultiDocumentSession call. Passing a valid BatchID value adds new source images to the session identified by the BatchID, and if a client is viewing that session, it pushes the new source images out to the client.
CallbackURL	String	(optional) URL that RTP Partner uses to call callback methods following annotation or approval events in this session. The default value is specified in **Callback > Callback URL**.

Output Parameter	Type	Description
Return	CreateMultiDocument SessionResult	Contains the following fields: • **BatchID**: a URL identifying the session • **Bootstrap**: a URL identifying the session. Note that both fields of this return type are populated with the same URL. For more information, see the description of the SessionReference output parameter in section 3.1.2 CreateSession.

Fault Codes		Description
		Client.FileNotFound Client.AccessForbidden Client.SyntaxError Server.InternalError Server.UnsupportedFileFormat Client.UnsupportedParameter

3.1.5 DestroySession

The DestroySession method is called to notify the service that the session is no longer necessary. After executing this method, the session URL cannot be used any more.

In SRS, sessions that are viewed in the client are destroyed when the client is closed. Sessions that are not opened by a client will eventually time out and be destroyed.

Input Parameter	Type	Description
SessionReference	String	The session URL exactly as returned by the respective CreateSession, CreateSession2, or CreateMultiDocumentSession method.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.SessionNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.1.6 EnumerateOpenSessions

The EnumerateOpenSessions method returns the list of sessions currently open in SRS.

Input Parameter	Type	Description
(none)		

Output Parameter	Type	Description
EnumerateOpenSessionsResult	SessionInfo[ ]	Each SessionInfo object contains the following fields: • **SessionInfo**[ ]: array of SessionInfo. • **SessionReference**: session URL. • **HostSessionID**: string identifier provided by caller of the session creation API. • **CTime**: date and time of session creation. • **Time**: date and time this session's information was added to the SessionInfoList. • **Source**: the source image files used to create the session. This will be empty if the session is a MultiDocumentSession. • **Active**: true if a client is viewing the session, false otherwise. *In SRS 1.0, this parameter will always be true.* • **NodeID**: *not used in SRS. Will contain the string ParameterNotSupportedInSRS* • **UserID**: ID of the user who initially created the session.

Fault Codes		Description
		Client.SyntaxError Server.InternalError

3.1.7 OnSessionClosed (Callback)

The OnSessionClosed method is an application defined callback method. It is called by SRS to notify the host application that session was closed. Currently in SRS, this callback is invoked when a session is destroyed via the DestroySession API, when the session has timed out, or when the client viewing the session closes the session.

Input Parameter	Type	Description
SessionReference	String	The session URL as returned by the session creation API.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
(none)		

3.2 Annotations

3.2.1 Overview

The most important change for RTP Partner users to be aware of in the Annotations, or Notes, APIs is that the NoteId field has changed from an integer value to a string value.

The table below indicates the mapping of SRS Note Types to RTP Note Types.

SRS Note Type	Persisted in SRS	Has Text in SRS	RTP Note Type
Text	Yes	Yes	Text
Pen	Yes	Yes	Freehand
Line	Yes	Yes	
Rectangle	Yes	Yes	Rect
Oval	Yes	Yes	Oval
Ruler	No	No	MeasureLine
Square Ruler	No	No	MeasureRect

Horizontal Guide	Yes	No	
Vertical Guide	Yes	No	
Densitometer	Yes (Optional)	No	Color
Multi-Shape	Yes	Yes	LooseGreasePen

The SRS Note Types that are not persisted (the gray shaded cells in the table) are not returned via the GetNotes and GetNotes2 APIs.GetNotes

The GetNotes method is called to synchronize the notes archive on the imaging server with the host application database.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
GetNotesResult	Note[ ]	An array of Note elements. The Note type contains the following fields • **Ver**: an integer value representing the version of the Note. *In SRS this value will always be 1; SRS does not version notes.* • **Id**: a string representing the unique identifier of the Note. ***Note:*** *This used to be an integer value in RTP.* • **Type**: enum value representing the type of Note. • **Page**: page number of where Note is located in Source. • **Time**: date and time when Note was last modified. • **Ctime**: date and time when Note was created. • **Author**: contains strings representing the UserId, First Name, and Last Name. • **Anchor**: anchor point in pixels of the upper left corner of the bounding box that encloses the Note graphic. This point is relative to the upper left corner of the page.

		• **PageSize**: size of page in pixels. *In SRS, this size value is an estimate for vector files (PDF, EPS). For PDF files, the size values are calculated based on 2400 dpi (RTP used 1200dpi). For multipage source files, the PageSize will be the size of the first page of the source file.* • **Text**: the text associated with the Note. • **Calibration**: in SRS this will always be undefined. *In SRS, this information is not persisted as a part of the Annotation data.*

Fault Codes		**Description**
		Client.FileNotFound Client.AccessForbidden Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.2.2 GetNotes2

The GetNotes2 method is called to synchronize the notes archive on imaging server with the host application database.

Input Parameter	**Type**	**Description**
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
SinceWhen	dateTime	All annotations with this modification time or later are returned. If the parameter is not used, all annotations stored for the source file are returned.

Output Parameter	**Type**	**Description**
GetNotes2Result	Note[ ]	An array of Note elements. See the GetNotes API for more details on the Note fields.

Fault Codes		Description
		Client.FileNotFound Client.AccessForbidden Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.2.3 DeleteNotes

The DeleteNotes method is called to delete notes from the SRS internal repository.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
NotesIDList	OperatedNoteId[ ]	An array of OperatedNoteID elements. Each OperatedNoteID object contains a Note ID field. The NoteID is returned by SRS in methods GetNotes, GetNotes2, and OnSynchronizeNotes (field ID). ***Note:*** *This used to be an integer value in RTP but is a string value in SRS.*

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.FileNotFound Client.NoteNotFound Client.AccessForbidden Client.SyntaxError

		Server.InternalError

3.2.4 OnSynchronizeNotes (Callback)

The OnSynchronizeNotes method is an application-defined callback function. It is called by the SRS Service to notify the host application of changes submitted by the Smart Review client to its internal annotations store.

Note: *SRS posts each annotation to the server as soon as it is created. That is, there is an implicit and immediate "synchronize" of every annotation. Therefore, the OnSynchronizeNotes callback will always contain only a single annotation in the OperatedNotesList. The parameter has been left as a list in order to preserve API compatibility with RTP Partner.*

Input Parameter	Type	Description
HostSessionID	String	A session identification string used by the host application. The service includes this ID in all the relevant reporting output.
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
Notes	OperatedNote[ ]	An array of OperatedNote elements. An OperatedNote object contains the following fields: • **Operation**: OperationCode enum (Added, Updated, Deleted) • **Ver**: an integer value representing the version of the Note. *In SRS this value will always be 1; SRS does not version notes.* • **Id**: a string representing the unique identifier of the Note. ***Note:*** *This used to be an integer value in RTP.* • **Type**: enum value representing the type of Note. • **Page**: page number of page where Note is located in the source. • **Time**: date and time when Note was last modified. • **Ctime**: date and time when Note was created.

		• **Author**: contains strings representing the UserId, First Name, and Last Name. • **Anchor**: anchor point, in pixels, of the upper left corner of the bounding box that encloses the Note graphic. This point is relative to the upper-left corner of the page. • **PageSize**: size of page in pixels. *In SRS this size value is an estimate for vector files (PDF, EPS). For PDF files, the size values are calculated based on 2400 dpi (RTP used 1200 dpi). For multipage source files, this PageSize is the size of the first page of the document.* • **Text**: the text associated with the Note. • **Calibration**: In SRS this will always be undefined. *In SRS this information is not persisted as a part of the Annotation data.*

Output Parameter	Type	Description
(none)		

Fault Codes	Type	Description
(none)		

3.3 Approval

3.3.1 Overview

Parameters Controlling Smart Review Client Approval Behavior

A number of variables are involved in controlling the behavior of callbacks and authentication when source approval is set in the Smart Review client:

- The callback URL can be specified in **Callback > Callback URL**, and/or in the Session Creation API call. The current SRS behavior is that the server setting is used as the base parameter, and it may be overridden by the value in the API call. If both values are blank, the callback will not be sent.
- The callback mode can be specified in **Approval > Enable Synchronous Callback for Approval**. It can be set to either synchronous or asynchronous, with the default being asynchronous. These two modes operate as follows:
    - In asynchronous mode, SRS calls the OnUserApproval callback after the approval transaction has been completed. The result of this callback does not affect the approval.
    - In synchronous mode, SRS calls SynchronizeUserApproval prior to completing the approval transaction. If this callback returns an error, the approval is not processed.
- The **Authorize On Approval** option can be selected in **Approval**. Optionally, it can be specified in a session creation API call, by adding the string "`, Approval/Authorize=on`" or "`, Approval/Authorize=off`" to the SessionType parameter; excluding the parameter defaults it to off. (This parameter was supported in RTP Partner although it was not documented in the API manual). The current SRS behavior is: if either the server setting or the session creation parameter is true, the flag is set to true. The behavior of this flag is as follows:
    - If **Authorize On Approval** is selected, the Smart Review client asks the user to provide a user name and password every time they approve a file. The user name and password is transmitted to the SRS server, which invokes the UserLogin callback to authorize the request. SRS does not authorize the request itself other than calling the appropriate callbacks.
    - If **Authorize On Approval** is not selected, the user is not prompted for a user name or password, and the UserLogin callback is not invoked.

Note: If any of these values are set via API call parameters, they override the values set as server parameters on the Configuration page.

The Smart Review client's Approval dialog box can be customized through the SetApprovalStatusList and SetDefaultApprovalStatusList methods. These methods determine the list of status values to be displayed to the user when approving a source file in Smart Review.

SRS's initial default approval status information is as follows:

SystemId	Status	Description	Read Only	Display Stamp	No Action
None	None	None	False	False	True
InProgress	InProgress	InProgress	True	False	False
Approved	Approved	Approved	False	True	False
ApprovedWith Changes	ApprovedWith Changes	ApprovedWith Changes	False	True	False
Request Changes	Request Changes	Request Changes	False	True	False

These defaults were taken from the RTP Partner.conf file.

Note: The SystemId value is not currently used.

3.3.2 GetApprovalStatusList

The GetApprovalStatusList method returns a list of available statuses and their descriptions for the source. If no statuses are defined, a default list of statuses is returned.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
GetApprovalStatus ListResult	GetApproval StatusListResult	Contains the following elements: • **Defined**: boolean • **Status**: array of **StatusDescription**. Valid if **Defined** is **True**. Each **StatusDescription** object includes the following information: • **SystemID**: a value of ApprovalSystemIDType. T*his parameter is not used.* • **Status**: the string stored by SRS when the source's approval status is set in the Smart Review client.

		• **Description**: the string that is displayed in the Smart Review client when the source's approval status is set to StatusDescription.Status • **ReadOnly**: boolean value to indicate that the Smart Review client should not display this StatusDescription.Description as an option in the client's Approval dialog box. • **DisplayStamp**: boolean value that indicates whether the document approval status is visible for the source file in the client if the overall document status for a source is set to this value. • **NoAction**: boolean value that indicates the client should not send a message to the server to update the source approval status if this status is selected in the client's Approval dialog box.

Fault Codes		**Description**
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.3 SetApprovalStatusList

The SetApprovalStatusList method overrides a default status list for a source file.

Invoking this API with the Params input parameter containing an empty array will remove the previously set approval status list for the specified source.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
Params	StatusDescription[ ]	An array of **StatusDescription**. See the GetApprovalStatusList API for a description of **StatusDescription** fields. An empty array will remove the previously set approval status list for the specified source.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.4 GetDefaultApprovalStatusList

The GetDefaultApprovalStatusList method returns a default list of statuses and their descriptions.

Input Parameter	Type	Description
(none)		

Output Parameter	Type	Description
GetDefault Approval StatusList	GetApprovalStatus ListResult	Contains the following elements: • **Defined**: boolean • **Status**: array of **StatusDescription**. Valid if **Defined** is **True**. See the GetApprovalStatusList API for a description of **StatusDescription** fields.

Fault Codes		Description
		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.5 SetDefaultApprovalStatusList

The SetDefaultApprovalStatusList method overrides the current default status list for the system. The updated default approval status information is stored in the SRS server's registry so this information will persist between SRS server restarts.

Invoking this API with the Params input parameter containing an empty array removes the previously set default approval status list for the system and the default approval status list will revert to the initial default settings described above.

Input Parameter	Type	Description
Params	StatusDescription[ ]	An array of **StatusDescription**. See the GetApprovalStatusList API for a description of **StatusDescription** fields. An empty array will remove the previously set default approval status list for the system and the default approval status list will revert to the initial default settings.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.6 GetUserApprovalStatus

The GetUserApprovalStatus method returns an approval status for the specified user and source file. Each user status is maintained independently on a per-source-file basis.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
UserID	String	Unique user identification in the host application. The UserID cannot contain spaces.

Output Parameter	Type	Description
GetUserApproval StatusResult	GetUserApproval StatusResult	Approval status per user. • **Defined**: boolean. If its value is **True**, the status field contains valid approval status for the user specified. • **Status**: string • **Comment**: string • **Updated**: dateTime. The last status update time. • **CalibrationStatus**: calibration of the monitor in the moment when the user's approval status has been changed.

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.7 SetUserApprovalStatus

The SetUserApprovalStatus method associates the approval status for the specified user and source file. Each user status is maintained independently on a per-source-file basis.

Invoking this API with the Status input parameter containing an empty string will remove any previously set approval status for the UserId and Source.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
UserID	String	Unique user identification in the host application. The UserID cannot contain spaces.
UserF	String	Human-readable user name.
UserL	String	
Status	String	New user approval status.
Comment	String	Comments for this transaction.
Calibration	String	Calibration of the monitor in the moment when the user's approval status has been changed.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.FileNotFound

		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.8 GetUsersApprovalStatus

The GetUsersApprovalStatus method returns a list of users and their statuses participating in the approval process for the source specified

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
GetUsersApproval StatusResult	UserStatus[ ]	An array of **UserStatus.** The UserStatus type contains the following fields: • **Author**: contains the UserId, and users' First and Last names. • **Updated**: the time that the approval was set • **Status**: set by the user. • **Comment**: comment string that was entered by the user when setting the status for the source. • **Calibration**: CalibrationStatus of the monitor on which the user set the status for the source.

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.9 GetDocumentApprovalStatus

The GetDocumentApprovalStatus method returns a document status for the specified Source.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
return	GetDocument ApprovalStatus Result	Document approval status. Contains the following elements: • **Defined**: boolean. If its value is **True**, the status field contains valid approval status for the user specified. • **Status**: string • **Updated**: dateTime. The last status update time.

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.10 SetDocumentApprovalStatus

The SetDocumentApprovalStatus method assigns an approval for the specified source. This approval displays in the Smart Review client when the source is viewed, provided the status is in the approval status list configured for the source and the DisplayStamp value for the status is True.

Invoking this API with the Status input parameter containing an empty string removes any previously set document approval status for the Source.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
Status	String	New document approval status.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.3.11 SetApproveOnClose

The SetApproveOnClose method modifies the approval-on-close flag for the session specified.

Input Parameter	Type	Description
SessionReference	String	The session URL exactly as returned by the respective Session Creation method.
ApproveOnCloseFlag	boolean	If set to TRUE, the Smart Review client will ask the user to set the approval status for the source being viewed before closing the application.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.SyntaxError Server.InternalError

		Client.UnsupportedParameter Client.SessionNotFound Client.InvalidForMultiDocSession

3.3.12 OnUserApproval (Callback)

The OnUserApproval method is an application-defined callback function. It is called by the SRS service to notify the host application about an approval status change made in the Smart Review client.

Note: OnUserApproval is called only when the callback is configured as asynchronous mode (that is, **Approval > Enable Synchronous Callback for Approval** is not selected).

Input Parameter	Type	Description
HostSessionID	string	A session identification string used by the host application.
Token	string	Login token as received previously from UserLogin. If UserLogin was not called (for example, if the **Authorize On Approval** option is not selected), this parameter will be blank.
Source	string	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
Status	string	New approval status.
Comment	string	Comments as inserted by the end-user in the Smart Review client.
Calibration	CalibrationStatus	Calibration of the monitor in the moment when the user approval status was changed.

Output Parameter	Type	Description
(none)		

Fault Codes	Type	Description
(none)		

3.3.13 SynchronizeUserApproval (Callback)

The SynchronizeUserApproval method is an application-defined callback function. It is called by the SRS service to notify the host application about approval status changes made in the Smart Review client.

Note: SynchronizeUserApproval is called only when **Approval > Enable Synchronous Callback for Approval** is selected.

During an approval transaction, SRS first calls the SynchronizeUserApproval callback, and sets the approval status only if this callback function returns a successful result.

Input Parameter	Type	Description
HostSessionID	string	A session identification string used by the host application.
Token	string	Login token as received previously from UserLogin. If UserLogin was not called (for example, if the **Authorize On Approval** option is not selected), this parameter will be blank.
Source	string	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
Status	string	New approval status.
Comment	string	Comments as inserted by the end-user into the Smart Review client.
Calibration	CalibrationStatus	Calibration of the monitor in the moment when the user approval status was changed.

Output Parameter	Type	Description
ErrorCode	String	Error code, received from the host application. An empty ErrorCode (“”) constitutes a successful result of the function. Any non-empty string constitutes a failure code. In this case, SRS does not update the approval status.
Description	String	If the returned ErrorCode is not empty (see the ErrorCode section), the Smart Review client displays this description in a message box.
Updated	dateTime	Time when approval status was granted by the

		host application. This value is saved in the SRS database.

Fault Codes	Type	Description
(none)		

3.3.14 UserLogin (Callback)

The UserLogin method is an application-defined callback function. It is called by the SRS service to authorize a client request for approval. Authorization is enabled through the **Approval > Authorize On Approval** parameter or by adding "`,Approval/Authorize= on`" to the SessionType parameter in the creation session API; this callback is only invoked if the **Authorize On Approval** option is selected.

Input Parameter	Type	Description
HostSessionID	string	A session identification string used by the host application.
UserID	string	User identification as entered by the user in the Smart Review client.
Password	string	User password as entered by the user in the Smart Review client.
Domain	string	SRS module sending the request. Approval workflow uses the domain as specified in **Callback > Callback Domain URL**.

Output Parameter	Type	Description
Return	string	Login token, if successful.

Fault Codes	Type	Description
(none)		

3.3.15 UserLogout (Callback)

The UserLogout method is an application-defined callback function. It is called by the SRS service to notify the host application that a session that previously invoked the UserLogin callback has either been destroyed or expired.

Input Parameter	Type	Description
HostSessionID	string	A session identification string used by the host application. The service includes this ID in all the relevant reporting output.
Token	string	Login token as received previously from UserLogin.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
(none)		

3.4 User Roles and Permissions

3.4.1 Overview

The permissions supported by SRS are as follows:

Permission	Hex Value	Integer Value
View	0x0002	2
Proof	0x0004	4
ViewNotes	0x0010	16
WriteNotes	0x0020	32
SaveAs	0x0040	64
Measure	0x0100	256
ReadApprove	0x0200	512
ChangeApprove	0x0400	1024

A user role defines a bitmask that enables a set of the permissions defined above. The bitmask is passed as an integer value to the SetUserRole or AdjustUserPermissions API. For example, if a user role grants the View, WriteNotes, and ReadApprove permissions, the integer permissions parameter will be 2 + 32 + 512 = 546. Once a user role is set, the role may be used in any of the Session Creation APIs to grant the session the permissions defined by the user role.

Any user role created via the SetUserRole API will automatically include the View permission.

Default user roles created by SRS are:

- Viewer: contains the View permission
- Reviewer: contains the View, Proof, ViewNotes, WriteNotes, SaveAs, Measure, ReadApprove, and ChangeApprove permissions

These default user roles can be modified by the SetUserRole and RemoveUserRole APIs or through **Roles**. If the default user roles are removed from the system, they will automatically be recreated the next time the SRS server is restarted.

The following table describes the functionality that each permission enables in the Smart Review client:

Permission	Description
Proof	• Enables the **File > Print to File** menu item • Enables the **File > Generate Proof Report** menu item
ViewNotes	• Enables the user to see all annotations made by other users on the source • Enables the **View > Show Annotations** and **Window > Annotation Manager** menu items (to view annotations made by any users)
WriteNotes	• Enables the user to add and view their own annotations on the source • Enables the annotation tools, except for the Densitometer, Ruler, and Square Ruler tools (to enable these tools, the Measure permission must also be selected for the role) • Enables the **View > Show Annotations** and **Window > Annotation Manager** menu items (to view their own annotations only)
SaveAs	Enables the **File > Print** menu item
Measure	Enables the Densitometer, Ruler, and Square Ruler annotation tools

	Note: To enable the user to check color density or take measurements using these tools, the WriteNotes permission must also be selected for the role.
ReadApprove	Displays other users' approval status for the source
ChangeApprove	Enables the user to click the **Approval** button and set their approval status for the current source

3.4.2 GetUserRole

The GetUserRole method returns the integer value of the Permission bitmask associated with the specified user role.

Input Parameter	Type	Description
UserRole	String	User role string identifier.

Output Parameter	Type	Description
GetUserRoleResult	Integer	Integer value of the associated Permissions bitmask.

Fault Codes		Description
		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.4.3 SetUserRole

The SetUserRole method defines a bitmask (a user role) for a set of permissions flags. User roles are persisted in the SRS server's database and can then be used with the Session Creation APIs to assign a set of permissions to a session.

Input Parameter	Type	Description
UserRole	String	User role string identifier
Permissions	Integer	Permissions mask. The mask can be a combination of the permissions defined in section 3.4 User Roles and Permissions.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.4.4 RemoveUserRole

The RemoveUserRole method deletes a previously defined bitmask.

Input Parameter	Type	Description
UserRole	String	User role string identifier

Output Parameter	Type	Description
RemoveUserRoleResult	Boolean	True, if the user role has been deleted.

Fault Codes		Description
		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.4.5 AdjustUserPermissions

The AdjustUserPermissions method changes the permissions bitmask for the session created previously by a Session Creation API.

Input Parameter	Type	Description
SessionReference	String	The session URL exactly as returned by the respective Session Creation method.
Permissions	Integer	New permissions bitmask for this session.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.SessionNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.4.6 QueryUserPermissions

The QueryUserPermissions method returns an effective permissions bitmask for the session specified.

Input Parameter	Type	Description
SessionReference	String	The session URL exactly as returned by the respective Session Creation method.

Output Parameter	Type	Description
QueryUserPermissionsResult	Integer	Effective permissions bitmask for this session.

Fault Codes		Description
		Client.SessionNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.5 Color

3.5.1 Overview

In SRS, color for a source file is managed by a color target. Color targets are created on the SRS server using the Color Target Manager. For more information about color management in SRS, see the *Smart Review System 1.0 Administration Guide*.

Note: It is possible to modify a color target on the SRS server after it has been associated with a source file. SRS will always the use the most recent version of a color target (at the time of client launch) when displaying the source file in the Smart Review client.

3.5.2 GetColorParameters2

The GetColorParameters2 method returns the color target name associated with the specified source file. If the SetColorParameters2 method has not been invoked for the specified source file, the system default color target name is returned. The system default color target can be configured by the SetDefaultColorParameters2 method or at **Configuration Settings > Color Management > Default Color Target**.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
GetColorParameters2Result	ColorParameters2	Contains the following fields: • **ProfileID**: the name of the color target associated with the specified source file. If the string is empty, there is no color target associated with the source. • **CMYKPrintingProcess**: *this field is not used by SRS.* • **ProofingCondition**: *this field is not used by SRS.* • **MonitorID**: *this field is not used by SRS.* • **ColorSync**: *this field is not used by SRS.* • **RGBQuality**: *this field is not used by SRS.* • **RGBType**: *this field is not used by SRS.*

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError

		Server.InternalError Client.UnsupportedParameter

3.5.3 GetDefaultColorParameters2

The GetDefaultColorParameters2 method returns the color target name currently configured as the system default. The system default color target can be configured by the SetDefaultColorParameters2 method or at **Configuration Settings > Color Management > Default Color Target**. The default color target name is persisted in the SRS server's registry.

Input Parameter	Type	Description
(none)		

Output Parameter	Type	Description
GetColorParameters2Result	ColorParameters2	See the GetColorParameters2 API description for ColorParameters2 field details.

Fault Codes		Description
		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.5.4 SetColorParameters2

The SetColorParameters2 method associates a color target name with the specified source files. The color target name must be created on the SRS server in the Color Target Manager before being passed as an input parameter to this API.

Invoking this API while one or more clients are viewing the specified source file will not push the new color information out to the open clients. The updated color information will take effect the next time a client is launched to view the source file.

Source file information can be viewed in the Smart Review client by selecting **Window > Element Info**. The image file's source profile(s) and color target information will be displayed.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
ProfileId	String	The name of the color target to associate with the source file.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.AccessForbidden Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.5.5 SetDefaultColorParameters2

The SetDefaultColorParameters2 method is used to establish the color target to be used as the SRS system default. This default value can also be configured at **Configuration Settings > Color Management > Default Color Target**. The default color target name is persisted in the SRS server's registry.

Input Parameter	Type	Description
ProfileId	String	The name of the color target to be used as the system default.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.AccessForbidden

		Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.6 File Management and Metadata

3.6.1 Overview

Source

The source is the key identifier of files within SRS. Specifically, the source is a directory path and file name combination relative to a configured Document Root, for files being added to the system (for example, through session creation APIs). After files have been added to the system, the relative file path is used for file identification and to establish the directory tree for SRS system files (for example, the cache).

3.6.2 Get File Info

The GetFileInfo method is called to read image information from the SRS service.

Width and height values for all pages of a multipage file are set to the width and height of the first page.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
GetFileInfoResult	PageInfo[ ]	Each PageInfo object contains the following elements: • **Page**: int. The page number. • **Width**: int. Image width in pixels. • **Height**: int. Image height in pixels.

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError

		Client.UnsupportedParameter

3.6.3 GetFileType

The GetFileType method returns the image file type as detected by the SRS service.

The following file types are supported in SRS:

- PostScript
- EPS
- PDF (up to Adobe® Acrobat® PDF version 8.0)
- JPG
- TIF
- DCS 1 and 2
- PSD
- AI (with PDF mode on)

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
GetFileTypeResult	GetFileTypeResult	Detected file type. Contains the following elements: • **RequiresRIP**: boolean. In SRS 1.0.0, **True** is always returned. • **Type**: string

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.6.4 GetThumbnail

The GetThumbnail method returns a thumbnail image of the specified source image file.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.
Page	Int	Page number of the page for which the thumbnail is required. Specifying Page = 0 for multipage source documents will cause this API to generate an animated .gif that contains all the pages of the source document.

Output Parameter	Type	Description
GetThumbnailResult	GetThumbnailResult	Contains the following fields : • **Width**: width of the thumbnail in pixels. • **Height**: height of the thumbnail in pixels. • **Type**: string indicating the type of image returned by this API (for example, **image/jpeg** or **image/bmp**). • **Image**: a base64Binary string of the thumbnail image.

Fault Codes		Description
		Client.FileNotFound Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.6.5 ObsoleteSource

The ObsoleteSource method is called to remove a particular file from the SRS system. As a result of the method execution, all information related to the old source (cache, annotations, and so on) is deleted from the system, and the file references are removed from the SRS database. This method does not have any effect on the host storage (for example, original files in the directories specified in **Document Roots > Default** or **Document Roots > Alternatives**).

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Client.FileNotFound Client.AccessForbidden Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.6.6 GetMetaData

The GetMetaData method returns metadata associated with the specified Source. To restore metadata, use the SetMetaData method.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

Output Parameter	Type	Description

Return	MetaData	Contains the following fields :
		• **Name** : string. The Source for which metadata is being obtained.
		• **Data**: a binary64 encoded string of the following information:
		  • Annotations
		  • User Approval Status
		  • Document Approval Status
		  • Approval Status List
		  • Color Parameters
		  • SRS version
		for the specified Source. Refer to Appendix A for a more detailed layout of the data returned by this API.
		• **Updated** : dateTime. The date and time that the information returned in the Data field was compiled.

Fault Codes		Description
		Client.FileNotFound Client.AccessForbidden Client.SyntaxError Server.InternalError Client.UnsupportedParameter

3.6.7 SetMetaData

The SetMetaData method transfers metadata associated with a Source back to the SRS server. A limitation of SRS 1.0.0 is that this method must only be invoked with metadata returned from the GetMetaData method.

Input Parameter	Type	Description
Source	String	Path to the source image file. The path is relative to either **Document Roots > Default** or one of the **Document Roots > Alternatives**.

MetaData	MetaData	Contains the following fields :
		• **Name** : string. The Source for which metadata is being obtained.
		• **Data**: a binary64 encoded string of the following information:
		  • Annotation
		  • User Approval Status
		  • Document Approval Status
		  • Approval Status List
		  • Color Parameters
		  • SRS version
		  for the specified Source. Refer to Appendix A for a more detailed layout of the data returned by this API.
		• **Updated** : dateTime. The date and time that the information returned in the Data field was compiled.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
(none)		

3.7 Server Utilities

3.7.1 GetVersionInfo

The GetVersionInfo method returns the versions of SRS components.

Input Parameter	Type	Description
(none)		
Output Parameter	**Type**	**Description**
GetVersionInfoResult	GetVersionInfoResult	Contains the following elements: **ServerVersion** – string **RIPs** – RIPInfoList **ColorTables** – an array of ColorTableInfo. Each element of the array contains the following fields : **ProfileId** – Generic Color Target Name **Version** – name of the most recent version of the generic Color Target Name **ProofingCondition** – *not used by SRS, set to the " ParameterNotSupportedInSRS" string.* **PrintingProcess** – *not used by SRS, set to the " ParameterNotSupportedInSRS" string.*
Fault Codes		**Description**
		Server.InternalError

3.7.2 GetServerInfo

The GetServerInfo method returns a server version, startup time, and the current system time.

Times are provided in local time, whereas RTP Partner provides times in GMT.

Input Parameter	Type	Description
(none)		

Output Parameter	Type	Description
GetServerInfoResult	GetServerInfoResult	Contains the following elements: • **Version**: string • **StartupTime**: dateTime (local time) • **SystemTime**: dateTime (local time)

Fault Codes		Description
		Server.InternalError

3.7.3 RestartServer

The RestartServer API will return *before* the SRS server has successfully restarted. This is due to the fact that restarting the SRS server requires a restart of the Microsoft Windows World Wide Web Publishing Service (w3svc.exe) to restart the SRS Control Interface and will require a few minutes of additional startup time.

The host application will not be able to access the SRS Control Interface until the World Wide Web Publishing Service is started on the SRS server.

Input Parameter	Type	Description
Message	String	Message that is logged to the diagnostics log before SRS is stopped.

Output Parameter	Type	Description
(none)		

Fault Codes		Description
		Server.InternalError

3.8 Data Dictionary

3.8.1 Simple Datatypes

Datatype	Type	Enumerated Values	Description
ApprovalSystemIDType	string	None InProgress Approved ApprovedWithChanges RequestChanges	Approval status types
CalibrationStatus	string	Undefined Calibrated Uncalibrated	Monitor calibration status
ColorSync	string	Undefined Off On	ColorSync support
NoteType	string	Rect Oval Color Text Freehand MeasureLine MeasureRect LooseGreasePen Line	Notes types supported by SRS
OperationCode	string	Added Updated Deleted	Operation being performed on note transferred from the Smart Review client

PlatformType	string	WinNT MacOS Win9x	Operating system type on top of which the Smart Review client runs
StreamType	string	JPEG RTI	Streaming protocol used during imaging *This parameter is not used in SRS.*

3.8.2 Complex Datatypes

Datatype	Type	Description
ColorParameters2	**ProfileID** – string **CMYKPrintingProcess** – string **ProofingCondition** – string **MonitorId** – string **ColorSync** – ColorSync **RGBQuality** – int **RGBType** – string	*The only parameter used by SRS is the ProfileId field. All other string parameters are set to "ParameterNot SupportedInSRS".* *ColorSync is set to Undefined and RGBQuality is set to 0.*
ColorTableInfo	**ProfileId** – string **Version** – string **ProofingCondition** – string **PrintingProcess** – string	*ProofingCondition and PrintingProcess are not used in SRS and are set to "ParameterNot SupportedInSRS".* *Version is the versioned name of the ColorTarget in the ProfileId field which has the form <colorTargetName>_ <date>*
CreateMultiDocumentSessionResult	**BatchID** – string **Bootstrap** – string	Returned value for the CreateMultiDocument SessionResult. Both parameters are set to the same Session URL.

Datatype	Type	Description
EnumerateOpenSessionsResult	**SessionInfo[ ]** – array of **SessionInfo**	
GetApprovalStatusListResult	**Defined** – boolean **Status** – array of **StatusDescription**	Returned value for the GetApprovalStatusList and GetDefaultApproval StatusList APIs
GetDocumentApprovalStatusResult	**Defined** – boolean **Status** – string **Updated** - dateTime	Return value for the GetDocumentApproval Status API
GetFileInfoResult	**PageInfo[ ]** – array of **PageInfo**	
GetFileTypeResult	**RequiresRIP** – boolean **Type** – string	Returned value for GetFileType API
GetNotesResult	**Note[ ]** – array of **Note**	
GetNotes2Result	**Note[ ]** – array of **Note**	
GetServerInfoResult	**Version** – string **StartupTime** – dateTime **SystemTime** – dateTime	Returned value for GetServerInfo API
GetThumbnailResult	**Width** – int **Height** – int **Type** – string **Image** – string base64Binary	
GetUserApprovalStatusResult	**Defined** – boolean **Status** – string **Comment** – string **Updated** – dateTime **CalibrationStatus** – CalibrationStatus	Returned value for GetUserApprovalStatus API

Datatype	Type	Description
GetVersionInfoResult	**ServerVersion** – string **RIPs** – RIPInfoList **ColorTables** – ColorTableInfo[ ]	Returned value for GetVersionInfo API
LaunchParams	**Viewer** – LaunchParamsViewer	Parameters for Smart Review client launch mechanism
LaunchParamsViewer	**Version** – string **URL** – string **VersionURL** – string	**Version**: required client version. T*his parameter is not needed in SRS and is set to "ParameterNot SupportedInSRS".* **URL**: imaging session access URL **versionURL**: URL to download the latest client software. *This parameter is not needed in SRS and is set to "ParameterNot SupportedInSRS"*
MetaDataList	**MetaData[ ]** – array of MetaData	
Note	**Ver** – int **Id** – string **Type** – NoteType **Page** – int **Time** – dateTime **Ctime** – dateTime **Author** – NoteAuthor **Anchor** – Wpoint **PageSize** – Wsize **Text** – string	Annotation ***Note:*** *Note.id used to be an integer value in RTP but is a string value in SRS.*

Datatype	Type	Description
	Calibration – CalibrationStatus	
NoteAuthor	**F** – string **L** – string **Aid** – string	Annotation Owner
OperatedNote	**Operation** – OperationCode **Ver** – int **Id** – int **Type** – NoteType **Page** – int **Time** – dateTime **Ctime** – dateTime **Author** – NoteAuthor **Anchor** – Wpoint **PageSize** – Wsize **Text** – string **Calibrated** – CalibrationStatus	Annotations transferred to host application
OperatedNoteID	**NoteID** – string	Unique string key, identifying annotation. Returned by SRS server in field **Id** (types **Note** and **OperatedNote**). ***Note:*** *This used to be an integer value in RTP but is a string value in SRS.*
PageInfo	**Page** – int **Width** – int	Source file page dimensions

Datatype	Type	Description
	Height – int	
PageInfoList	**PageInfo[ ]** – array of PageInfo	
RIPInfo	**RIPId** – string **Version** – string	RIP version, as installed on the server
RIPInfoList	**RIPInfo[ ]** – array of RIPInfo	
SessionInfo	**SessionReference** – string **HostSessionID** – string **CTime** – dateTime **Time** – dateTime **Source** – string **Active** – bool **NodeID** – string **UserID** – string	
SessionInfoList	**SessionInfo[ ]** – array of SessionInfo	
StatusDescription	**SystemID** – ApprovalSystemIDType **Status** – string **Description** – string **ReadOnly** – bool **DisplayStamp** – bool **NoAction** – bool	Defines the parameters of a single approval status displayed to users in the Smart Review client's Approval dialog box
UserStatus	**Author** – NoteAuthor **Updated** – dateTime **Status** – string	Approval user status description

Datatype	Type	Description
	Comment – string **Calibration** – CalibrationStatus	
WPoint	**Xcoord** – float **Ycoord** – float	Point definition
WSize	**Xsize** – float **Ysize** – float	Size definition

3.8.3 Exceptions

Fault Code	Description
Client.FileNotFound	The original image file has not been found on the host storage.
Client.SessionNotFound	The session object was not found for specified session reference.
Client.AccessForbidden	The attempt has been made to access the image file and its attached information (such as annotations, approval, etc.), but the access was denied due to incompatible user rights.
Client.SyntaxError	The SOAP request is not valid or not recognized as such by the Partner service.
Client.InvalidForMultiDocSession	The action is not valid for a multidocument session reference.
Server.InternalError	The server detected an internal error during processing of client request.
Server.UnsupportedFileFormat	The requested file format is not supported.
Client.UnsupportedParameter	The server cannot accept the parameter and/or its value.
Client.ColorTableError	The server could not access a specified color profile.
Client.NoteNotFound	Attempt to delete annotation that has not been found on the annotations store.

4. APPENDIX A: METADATA STRUCTURE

The GetMetaData method returns the following structure:

```
<GetMetaDataResult>
        <Name>string</Name>
        <Data>base64Binary</Data>
        <Updated>dateTime</Updated>
</GetMetaDataResult>
```

The content of the <Data> element is a text string that is base64 encoded. Decoding this string will result in the following XML structure.

```
<?xml version="1.0" encoding="utf-16" ?>

<methodResponse>

  <params>

    <param>

      <value>

        <struct>

          <member>

            <name>pageAnnotations</name>

            <value> …. </value>

          </member>

          <member>

            <name>approvalStatusList</name>

            <value> …. </value>

          </member>

          <member>

            <name>srsServerVersion</name>

            <value> …. </value>

          </member>

          <member>

            <name>colorParameter</name>

            <value> …. </value>

          </member>

          <member>

            <name>docApproval</name>

            <value> …. </value>

          </member>
```

```
        <member>
          <name>usersApproval</name>
          <value> …. </value>
        </member>
      </struct>
    </value>
  </param>
 </params>
</methodResponse>
```

4.1 Page Annotations

All position information in SRS annotation metadata is normalized to be a unit-less value between 0 and 1.

The following example illustrates the structure of SRS metadata pageAnnotations.

```
<name>pageAnnotations</name>
<value>
 <array>
  <data><value><struct>
   <member>
    <name>annotationsOnPage</name>
    <value>
     <array>
      <data><value><struct>
       <member>
        <name>type</name>
        <value>
         <string>Text</string>
        </value>
       </member>
       <member>
        <name>sessionId</name>
         <value>
          <string>25214014BFC6234A9B9F527813C609B3</string>
         </value>
       </member>
       <member>
        <name>lightColor</name>
```

```
        <value>
          <struct>
            <member>
              <name>green</name>
              <value>
                <int>0</int>
              </value>
            </member>
            <member>
              <name>blue</name>
              <value>
                <int>0</int>
              </value>
            </member>
            <member>
              <name>red</name>
              <value>
                <int>180</int>
              </value>
            </member>
          </struct>
        </value>
      </member>
      <member>
        <name>position</name>
        <value>
          <struct>
            <member>
              <name>x</name>
              <value>
                <string>0.422</string>
              </value>
            </member>
            <member>
              <name>y</name>
              <value>
                <string>0.562271062271062</string>
```

```
      </value>
     </member>
    </struct>
  </value>
</member>
<member>
  <name>annotApprovalStatusComment</name>
  <value>
    <string>NoAnnotText</string>
  </value>
</member>
<member>
  <name>darkColor</name>
  <value>
    <struct>
     <member>
       <name>green</name>
       <value>
         <int>0</int>
       </value>
     </member>
     <member>
       <name>blue</name>
       <value>
         <int>0</int>
       </value>
     </member>
     <member>
       <name>red</name>
       <value>
         <int>180</int>
       </value>
     </member>
    </struct>
  </value>
</member>
<member>
```

```
  <name>userName</name>
  <value>
    <string>AJ Leitch (user100)</string>
  </value>
</member>
<member>
  <name>timeLastModified</name>
  <value>
    <string>1193699039734</string>
  </value>
</member>
<member>
  <name>parentAnnotationId</name>
  <value>
    <string>-1</string>
  </value>
</member>
<member>
  <name>annotApprovalStatus</name>
  <value>
    <string>Created</string>
  </value>
</member>
<member>
  <name>relativePosition</name>
  <value>
    <struct>
      <member>
        <name>x</name>
        <value>
          <string>0</string>
        </value>
      </member>
      <member>
        <name>y</name>
        <value>
          <string>0.0274725274725275</string>
```

```
                </value>
               </member>
              </struct>
             </value>
            </member>
            <member>
             <name>timeCreated</name>
             <value>
              <string>1193699039734</string>
             </value>
            </member>
            <member>
             <name>id</name>
             <value>
              <string>AJ Leitch (user100)1193699039734</string>
             </value>
            </member>
            <member>
             <name>text</name>
             <value>
              <string>Just some text</string>
             </value>
            </member>
           </struct></value></data>
         </array>
        </value>
      </member>
      <member>
       <name>pageNumber</name>
       <value>
        <int>1</int>
       </value>
      </member>
    </struct></value></data>
  </array>
</value>
```

Essentially this structure indicates that there is an array of annotations for each page of a source element. The information for each annotation is stored internally as a hash table, that is, a set of name/value pairs.

Keys common to all annotation types:

Key name	Value
type	String: one of text, line, rectangle, oval, densities, multi-shape or guide
sessionId	String—an SRS internal GUID
lightColor	A hash table representing an RGB color value
annotApprovalStatusComment	String—value should be “NoAnnotText”
darkColor	A hash table representing an RGB color value
userName	String—FirstName LastName (UserId)
timeLastModified	String
parentAnnotationId	String—value is normally -1. The only time it is not is if the annotation is a child of a multi-shape annotation.
annotApprovalStatus	String—value should be “Created”
timeCreated	String
Id	String—a unique identifier for the annotation. The form appears to be username(GUID). If the annotation is a multi-dhape annotation, the id will be “parent_userName(GUID)”. If the annotation is a child of a multi-shape annotation, the id will be “childX_userName(GUID)” where X is an integer value 1, 2, 3, etc and the GUID value is the same as the parent multi-shape annotation.

Other key names that may appear in an annotation structure are defined in the following table:

Key Name	Annotation Type Applied To	Value
relativePosition	Text, Line, Rect, Oval, Multi-Shape	A hash table representing the X,Y coordinate of the annotation’s text box relative to the end point of the annotation
lineType	Line	String—value can either be line_line

		or pen_line
guideType	Guide	String—value can either be vertical_guide or horizontal_guide
Position	Text, Guide, Densities	A hash table representing the X,Y coordinate of the annotation
Text	Text, Line, Rect, Oval, Multi-shape	String—the text associated with the annotation. If for some reason the annotation doesn't have any text, then value should be "NoAnnotText".
Densities	Densities	Array of 4 hash tables, each hash table has two key/value pairs—keys are Name and Value. The Name value is either C, M, Y or K, and Value is a string representation of an integer value between 0 and 255.
Bounds	Rect, Oval	A hash table with 4 key/value pairs. The key names are: width, height, x and y. The value associated with each key is a string version of a float value between 0.0 and 1.0. These values represent the extents of the Rect or Oval and the coordinate of the upper left corner of the enclosing box.
Points	Line (either pen or line)	An array of Position hash tables representing the points in a line.
childAnnotations	Multi-Shape	An array of Line, pen_line annotation hash tables. The parentId value of each child annotation should be the id value from the multi-shape annotation that contains this childAnnotations key. The id value of each child annotation will be "childX_userId(GUID)" where X is an integer value 1, 2, etc.

The following sample is the annotation data for a source element without any annotations.

```
<name>pageAnnotations</name>
<value>
  <array>
    <data>
      <value>
        <struct>
          <member>
            <name>annotationsOnPage</name>
              <value>
                <array>
                  <data />
                </array>
              </value>
            </member>
            <member>
              <name>pageNumber</name>
              <value>
                <int>1</int>
              </value>
            </member>
          </struct>
        </value>
      </data>
    </array>
</value>
```

4.2 Approval Status List

The Approval Status List is an array of hash tables. An Approval Status List entry will contain the following key/value pairs.

Key Name	Value
NoAction	Boolean
Status	String
Description	String
SystemID	String
ReadOnly	Boolean
DisplayStamp	Boolean

The following sample of ApprovalStatusList metadata contains two approval status values:

```
<name>approvalStatusList</name>
<value>
  <array>
    <data>
      <value>
        <struct>
          <member>
            <name>NoAction</name>
            <value>
              <boolean>0</boolean>
            </value>
          </member>
          <member>
            <name>Status</name>
            <value>
              <string>Request Changes</string>
            </value>
                                                  </member>
          <member>
            <name>Description</name>
            <value>
              <string>Request Changes</string>
            </value>
```

```
        </member>
        <member>
          <name>SystemID</name>
          <value>
            <string>RequestChanges</string>
          </value>
        </member>
        <member>
          <name>ReadOnly</name>
          <value>
            <boolean>0</boolean>
          </value>
        </member>
        <member>
          <name>DisplayStamp</name>
          <value>
            <boolean>1</boolean>
          </value>
        </member>
      </struct>
    </value>
    <value>
      <struct>
        <member>
          <name>NoAction</name>
          <value>
            <boolean>0</boolean>
          </value>
        </member>
        <member>
          <name>Status</name>
          <value>
            <string>Approved With Changes</string>
          </value>
        </member>
        <member>
          <name>Description</name>
```

```
          <value>
            <string>Approved With Changes</string>
          </value>
        </member>
        <member>
          <name>SystemID</name>
          <value>
            <string>ApprovedWithChanges</string>
          </value>
        </member>
        <member>
          <name>ReadOnly</name>
          <value>
            <boolean>0</boolean>
          </value>
        </member>
        <member>
          <name>DisplayStamp</name>
          <value>
            <boolean>1</boolean>
          </value>
        </member>
      </struct>
    </value>
  </data>
 </array>
</value>
```

4.3 SRS Server Version

This is the version of the SRS server at the time the metadata was retrieved for the source element.

```
<name>srsServerVersion</name>
<value>
   <string>1.0.0.394</string>
</value>
```

4.4 Color Parameter

The color information for the source element. The ProfileID key/value pair contains the data that links a color target name configured on the SRS server to a source element. The remaining key/value pairs are legacy parameters from RTP Partner.

```
<name>colorParameter</name>
  <value>
    <struct>
      <member>
        <name>CMYKPrintingProcess</name>
        <value>
          <string>ParameterNotSupportedInSRS</string>
        </value>
      </member>
      <member>
        <name>MonitorID</name>
        <value>
          <string>ParameterNotSupportedInSRS</string>
        </value>
      </member>
      <member>
        <name>RGBQuality</name>
        <value>
          <int>0</int>
        </value>
      </member>
      <member>
        <name>ProfileID</name>
        <value>
          <string>TestColorTarget</string>
        </value>
      </member>
      <member>
        <name>ColorSync</name>
        <value>
          <string>UserDefined</string>
        </value>
      </member>
```

```
      <member>
        <name>ProofingCondition</name>
        <value>
          <string>ParameterNotSupportedInSRS</string>
        </value>
      </member>
    <member>
      <name>RGBType</name>
      <value>
        <string>ParameterNotSupportedInSRS</string>
      </value>
    </member>
  </struct>
</value>
```

Note that if there is no color target set for the system (or file), then the format of the ProfileId is as follows:

```
  <name>ProfileID</name>
  <value>
    <string />
  </value>
```

4.5 Document Approval

The following is an example of the Document Approval section of the SRS metadata structure. This example is for a source element that has had SetDocumentApprovalStatus invoked.

```
<name>docApproval</name>
  <value>
    <struct>
      <member>
        <name>Status</name>
        <value>
          <string>Approved</string>
        </value>
      </member>
      <member>
        <name>Defined</name>
```

```
      <value>
        <boolean>1</boolean>
      </value>
    </member>
   <member>
    <name>Updated</name>
    <value>
      <string>10/29/2007 15:56:09</string>
    </value>
   </member>
  </struct>
</value>
```

The next sample is for a source element that does NOT have a Document Approval Status set.

```
<name>docApproval</name>
  <value>
    <struct>
      <member>
        <name>Status</name>
        <value>
          <string>Status Undefined</string>
        </value>
      </member>
      <member>
        <name>Defined</name>
        <value>
          <boolean>0</boolean>
        </value>
      </member>
      <member>
        <name>Updated</name>
        <value>
          <string>11/08/2007 10:08:01</string>
        </value>
    </member>
```

```
  </struct>
</value>
```

4.6 Users Approval Status

The following is a sample of SRS user approval status metadata. This example has approval data for three users.

```
<name>usersApprovals</name>
<value>
  <array>
    <data>
      <value>
        <struct>
          <member>
            <name>Status</name>
            <value>
              <string>RequestChanges</string>
            </value>
          </member>
          <member>
            <name>Calibration</name>
            <value>
              <string>Uncalibrated</string>
            </value>
          </member>
          <member>
            <name>Author</name>
            <value>
              <struct>
                <member>
                  <name>F</name>
                  <value>
                    <string>AJ</string>
                  </value>
                </member>
                <member>
                  <name>Aid</name>
                  <value>
```

```
            <string>User100</string>
          </value>
        </member>
        <member>
          <name>L</name>
          <value>
            <string>Leitch</string>
          </value>
        </member>
      </struct>
    </value>
  </member>
  <member>
    <name>Updated</name>
    <value>
      <string>10/29/2007 15:56:08</string>
    </value>
  </member>
  <member>
    <name>Comment</name>
    <value>
      <string />
    </value>
  </member>
 </struct>
</value>
<value>
 <struct>
  <member>
    <name>Status</name>
    <value>
      <string>Rejected</string>
    </value>
  </member>
  <member>
    <name>Calibration</name>
    <value>
```

```
      <string>Undefined</string>
    </value>
  </member>
  <member>
    <name>Author</name>
    <value>
      <struct>
        <member>
          <name>F</name>
          <value>
            <string>Test</string>
          </value>
        </member>
        <member>
          <name>Aid</name>
          <value>
            <string>User555</string>
          </value>
        </member>
        <member>
          <name>L</name>
          <value>
            <string>Person_1</string>
          </value>
        </member>
      </struct>
    </value>
  </member>
  <member>
    <name>Updated</name>
    <value>
      <string>10/29/2007 15:56:12</string>
    </value>
  </member>
  <member>
    <name>Comment</name>
    <value>
```

```
          <string>blah blah</string>
        </value>
      </member>
    </struct>
  </value>
  <value>
    <struct>
      <member>
        <name>Status</name>
        <value>
          <string>ApprovedWithChanges</string>
        </value>
      </member>
      <member>
        <name>Calibration</name>
        <value>
          <string>Undefined</string>
        </value>
      </member>
        <member>
          <name>Author</name>
          <value>
          <struct>
            <member>
              <name>F</name>
              <value>
                <string>Test</string>
              </value>
            </member>
            <member>
              <name>Aid</name>
              <value>
                <string>User666</string>
              </value>
            </member>
            <member>
              <name>L</name>
```

```
            <value>
              <string>Person_2</string>
            </value>
          </member>
        </struct>
      </value>
    </member>
    <member>
      <name>Updated</name>
      <value>
        <string>10/29/2007 15:56:13</string>
      </value>
    </member>
    <member>
      <name>Comment</name>
      <value>
        <string>yakity yakity yak</string>
      </value>
    </member>
  </struct>
 </value>
 </data>
</array>
</value>
```

The next sample is for a source element that does not have any user approvals set.

```
<name>usersApprovals</name>
<value>
  <array>
    <data />
  </array>
</value>
```