Getting Started with the Java API

Applies to

RightFax 9.3, 9.4, 10.0.0, 10.5, 10.6, 16.2, 16.4, 16.6

Summary

In RightFax, a Java API is available, and this article provides the information necessary to get started with the Java API.

Resolution

Getting Started

The RightFax API for Java converts Java to XML on the host computer and then transmits it to the RightFax Integration Module. The RightFax XML Interface converts the XML to FCL.

The RightFax Integration Module can then process and send the document from the RightFax server. The RightFax API for Java can submit an outbound document, query the RightFax server for the status of a document, and perform actions (forward, delete, or create a library document) on previously sent documents.

Disclaimer

This article is provided as a courtesy. Information in this article is believed to be accurate; however, it is provided "as is", without any guarantee of accuracy and without warranty of any kind, express or implied, of merchantability or fitness for any particular purpose.

Requirements

- Your server must be licensed with the RightFax Integration Module license.
- Sun Microsystems, Inc. Java Developer's Kit version 1.1.8 or later.

- Internet Information Server (IIS) version 6.0 or later installed on the RightFax server.
- CGI modules and ISAPI modules enabled in IIS.
- Parent Paths must also be enabled, either for the Default Web Site as a whole, or for the RFXML virtual directory which appears after the installation of the Java/XML API.
- RFXML Website installed. During installation of RightFax, or in Programs and Features (formerly Add or Remove Programs), customize the RightFax Product Suite installation. Under Expand Server Module, choose Java/XML API. For additional details, see the RightFax installation Guide.

Java API for RightFax Class Files

The RightFax Java API class files can be found in &#60RightFax Install Directory&#62\RightFax\Production\xml\java\RFJavaInt.zip. The class files should be zipped into a folder with the name of your choice, imported into your Java project, and referenced. Example: import RightFAX.*;

Java API Code Examples

Submitting a fax document

To submit an outbound document, you must set sender info, recipient info, and add the content for the primary document (if any). These are encapsulated in an RFDoc object, in the RFaxSubmit object. Then you can add any attachments. Here is a short description of how to do this. Note that these steps are just the basics.

1. Create an RFaxSubmit object.
2. Set the target URL for the RightFAX server, using setTargetURL method. This takes a String, NOT a URL object (it creates the URL object).
3. Use the RFDoc object, called m_FaxDocument, in the RFaxSubmit object to set all the non-attachment information about the document.
    - Call m_FaxDocument.setStyleSheet to set the style sheet for the document.
    - Call m_FaxDocument.setXMLNS to specify the XML namespace.
    - Call m_FaxDocument.setSenderInfo (…) to set the information about the person sending the fax. This method has been overloaded, with a version that takes all information as parameters, a version that only takes the minimum information (RFUser ID), and versions that take the more common information. If there is not an overloaded version with exactly what you want, use one that takes more, and set the parameters you do not want to use to empty strings ("").
    - Call m_FaxDocument.addRecipient (…) for each recipient. Each time you call this, a recipient adds to the list. This method has been overloaded, with a version that takes all

information as parameters, a version that only takes the minimum information (Fax Number), and versions that take the more common information. If there is not an overloaded version with exactly what you want, use one that takes more, and set the parameters you do not want to use to empty strings ("").

- Call m_FaxDocument.setBody (…) to set the contents of the fax body (if any). If the body data is not raw text, make sure to set the type as well.
- Call m_FaxDocument.setCoverText (…) to set the contents of the fax cover page text (if any). If the cover text is a data type other than TXT, then use m_FaxDocument.setCoverTextType (…) to set the type.

4.If you have any attachments, make a call to addAttachment (…) for each attachment. Each time you call this, an attachment is added to the list. This method takes a fully qualified path as its parameter, not the contents of the file to be attached.

5. Once all the data is set, you call the submit method. You receive a Vector of RFStatus objects. Each object contains the status for one recipient, saying whether it was passed on for delivery, or there was an error. Note: You can call submitEx method in place of submit. This returns a String containing the unparsed XML returned by the RightFax server. The XML contains the status of each recipient, and is in the form of XML_FAX_SUBMIT_REPLY.xml example.

```
import RightFAX.*; import java.net.MalformedURLException; import java.net.UnknownHostException; import java.io.IOException; import java.util.*; class FaxSubmit { //Create a outbound fax object RFaxSubmit obFS = new RFaxSubmit(); //Set the URL of the RightFAX server obFS.setTargetURL("http://www.company.com/"); //Set the stylesheet and schema paths obFS.m_FaxDocument.setStyleSheet("XML_FAX_SUBMIT.xsl"); obFS.m_FaxDocument.setXMLNS("x-schema:C:\\Program Files (x86)\\RightFax\\Production\\xml\\schemas\\XML_FAX_SUBMIT_schema.xml"); //Set the information on who is sending the fax obFS.m_FaxDocument.setSenderInfo("Bill Smith", "", "Acme, Co.", "", "", "", "", "", "", "BillS"); //Add 2 recipients try { obFS.m_FaxDocument.addRecipient("PRODXML:0001", "555-1234", "", "Jim Jackson", "", "", "", "", "", "", "", ""); } catch(RFNoFaxNumberException nfne) { System.out.println(nfne.toString()); } catch(RFInvalidIDException iide) { System.out.println(iide.toString()); } //Add an email and a printer recipient try { obFS.m_FaxDocument.addRecipient_email("EMAIL:00000001", "smithj@company.com", "", "Here is the document", ""); obFS.m_FaxDocument.addRecipient_printer("PRNT:000000001", "MyPrinter", (short)1); } catch(RFNoDestinationException nde) { System.out.println(nde.toString()); } catch(RFInvalidIDException iide) { System.out.println(iide.toString()); } //Set the body text obFS.m_FaxDocument.setBody("Here is some body text","TXT", -1, -1, -1, "Arial", -1, -1, 10); //Set the cover text obFS.m_FaxDocument.setCoverText("Here is some cover text"); //Add attachments obFS.addAttachment("c:\\documents\\mydoc.doc"); obFS.addAttachment("c:\\documents\\license.pdf"); //Send the document, and get back the results. Vector obRetList = null; try { obRetList = obFS.submit(); } catch(MalformedURLException mue) { System.out.println(mue.toString()); } catch(UnknownHostException uhe) {
```

```
System.out.println(uhe.toString()); } catch(IOException ioe) { System.out.println(ioe.toString()); }
catch(RFNoDataException nde) { System.out.println("Error:" + nde.toString()); } //Output the results int
nSize = obRetList.size(); for (int i = 0; i < nSize; i++) { RFStatus obStat = (RFStatus)(obRetList.get(i));
System.out.println((i+1) + "-"); System.out.println("\tID: " + obStat.getID());
System.out.println("\tStatusCode: " + obStat.getStatusCode()); System.out.println("\tStatusMessage: " +
obStat.getStatusMsg()); } }
```

Adding attachments to a fax document

When including attachments with a fax, the location of the files is expressed relative to where your java code is executed, as in the following example statements:

```
// Add a document from a local directory on your Microsoft Windows client
obFS.addAttachment("C:\\java\\attach\\attachment.html"); // Add an attachment stored in a shared
network directory
obFS.addAttachment("\\\\myFileStore.myDomain.com\\attachments\\myAttachment.pdf");
obFS.addAttachment("\\\\192.168.0.123\\attachments\\myAttachment.html");
```

Including binary file contents in the fax body

Fax documents created with the Java API for RightFax are passed to the RightFax Server as XML. File contents included in the XML must be in a format that can be processed by the XML parser. In this case, encode your files as BASE64, and insert them in the body of the document like the following example:

```
// Base64 encode document with encode64() // Insert in the fax body with setBody() // Set the body type
to BASE64 with setBodyEncoding() try {
obFS.m_FaxDocument.setBody(obRFE.encode64_File("\\\\10.104.109.236\\attach\\attachment.html")," 
html"); obFS.m_FaxDocument.setBodyEncoding(1); } catch(FileNotFoundException fnfe) {
System.out.println(fnfe.toString()); } catch(IOException ioe) { System.out.println(ioe.toString()); }
```

Querying fax documents

To perform a query on the status of a fax (or group of faxes), you must set the criteria for each query you want to do. Here is a short description of the basics.

- Create an RFaxQuery object.
- Set the target URL for the RightFAX server, using setTargetURL method. This takes a String, NOT a URL object (it creates the URL object).

- Call addQuery (…) for each query you want to add to the request. This method has been overloaded, with a version that takes all information as parameters, and versions that take the more common information. If there is not an overloaded version with exactly what you want, use one that takes more, and set the parameters you do not want to use to empty strings (""), or null for Calender objects. NOTE: You must set at least one query criteria parameter.
- Once all the queries are set, you call the submit method. You receive a Vector of RFStatus objects. Each object contains the status of one fax (not one query (a query could return many fax statuses)).

```
import RightFAX.*; import java.net.MalformedURLException; import java.net.UnknownHostException; import java.io.IOException; import java.util.*; class FaxQuery { //Create a RFaxQuery object RFaxQuery obQ = new RFaxQuery(); //Set the URL of the RightFAX server obQ.setTargetURL ("http://www.company.com/""); //Criteria for one query. try { obQ.addQuery ("PRODXML:0001", null, null, "", "", ""); } catch (RFEmptyQueryException eqe) { System.out.println (eqe.toString(); } catch (RFInvalidIDException iide) { System.out.println (iide.toString()); } //Send the query, and get back the results Vector obQRetList = null; try { obQRetList = obQ.submit(); } catch (MalformedURLException mue) { System.out.println (mue.toString()); } catch (UnknownHostException uhe) { System.out.println (uhe.toString()); } catch (IOException ioe) { System.out.println (ioe.toString()); } catch (RFNoDataException nde) { System.out.println (nde.toString()); } //Output the results int nQSize = obQRetList.size(); for (int i = 0; i < nQSize; i++) { RFStatus obStat = (RFStatus)(obQRetList.get(i)); System.out.println ((i+1) + "-"); System.out.println ("\tID: " + obStat.getID()); System.out.println ("\tStatusCode: " + obStat.getStatusCode()); System.out.println ("\tStatusMessage: " + obStat.getStatusMsg()); } }
```

Fax Actions

To perform an action on a fax, you require the unique_id for that fax. Once you have that, you can delete the fax, forward it to another fax number, or use it to create a library document. If you do not have the unique_id, you can obtain it by doing a query on the information that you do have (see above). Here is a short description of how to do this. Note that these steps are just the basics.

- Create an RFaxAction object.
- Set the target URL for the RightFAX server, using setTargetURL method. This takes a String, NOT a URL object (it creates the URL object).
- Call addForwardAction (…) for each forward action you want to perform. This method has been overloaded, with a version that take all information as parameters, a version that only take the minimum information (ID and Fax Number). If there is not an overloaded version with exactly what you want, use one that takes more, and set the parameters you do not want to use to empty strings ("").
- Call addDeleteAction (…) for each delete action you want to perform The ID parameter is required.

- Call addLibDocAction (…) for each create library document action you want to perform. The ID, LibDocID, and LibDocDescription parameters are all required.
- Once all the actions are set, you call the submit method. You receive a Vector of RFStatus objects. Each object contains the status of one action showing whether or not the action was successful.

```
import RightFAX.*; import java.net.MalformedURLException; import java.net.UnknownHostException; import java.io.IOException; import java.util.*; class FaxAction { //Create a RFaxAction object RFaxAction obA = new RFaxAction(); //Set the URL of the RightFAX server obA.setTargetURL ("http://www.company.com/"); //Create a forward action try { if (!obA.addForwardAction("PRODXML:0001", "555-6789", "Mike Michell", "", "", "", "")) { System.out.println ("Add Forward Action Failed"); } } catch (RFNoFaxNumberException nfne) { System.out.println (nfne.toString()); } catch (RFNoIDException nide) { System.out.println (nide.toString()); } catch (RFInvalidOpException ioe) { System.out.println (ioe.toString()); } //Create a delete action try { if (!obA.addDeleteAction("PRODXML:0002")) { System.out.println ("Add Delete Action Failed"); } } catch (RFNoIDException nide) { System.out.println (nide.toString()); } catch (RFInvalidOpException ioe) { System.out.println (ioe.toString()); } //Send the action requests, and get back the results Vector obARetList = null; try { obARetList = obA.submit(); } catch (MalformedURLException mue) { System.out.println (mue.toString()); } catch (UnknownHostException uhe) { System.out.println (uhe.toString()); } catch (IOException ioe) { System.out.println (ioe.toString()); } catch (RFNoDataException nde) { System.out.println (nde.toString()); } //Output the results int nASize = obARetList.size(); for (int i = 0; i < nASize; i++) { RFStatus obStat = (RFStatus)(obARetList.get(i)); System.out.println ((i+1) + "-"); System.out.println ("\tID: " + obStat.getID()); System.out.println ("\tStatusCode: " + obStat.getStatusCode()); System.out.println ("\tStatusMessage: " + obStat.getStatusMsg()); } }
```

Advantage Technologies
228 East 45th Street
4th Floor South
New York, NY 10017
866-730-1700
info@atechnologies.com
www.atechnologies.com

About Advantage

Advantage Technologies has been providing on-premise and cloud-based enterprise fax and automated electronic document delivery solutions for over 20 years. Our team has completed thousands of successful system deployments worldwide in such industries as finance, insurance, banking, government, manufacturing, transportation, and healthcare.

Our North American helpdesk and sales team are certified on OpenText RightFax, Alchemy, RightFax Connect, Secure Mail, Secure MFT, Brooktrout fax boards and FoIP software, Dialogic Media Gateways, Sonus Fax Gateways, and cloud-based fax solutions. Advantage Technologies is a leading OpenText Platinum Partner and Authorized Support Partner (ASP).

Throughout our partnership with OpenText, Advantage has been recognized as RightFax Partner of the Year, IX Partner of the Year, and an IX Partner Leader.

opentext™ | Partner Support

opentext™ | Partner Distributor

opentext™ | Partner Reseller Platinum